# Development of Explicit Finite Difference-Based Simulation System for Impact Studies

*by*

Shaw Voon Wong, B.Eng.

A Thesis submitted in fulfillment of the requirement for the degree of

Doctor of Philosophy

Supervisors:

Professor M.S.J. Hashmi

Dr. A.M.S. Hamouda

Dublin City University

School of Mechanical & Manufacturing Engineering

June 2000

To

my grandmother

&

my parents

# Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: [illegible] ID No: 97970514

Wong, Shaw Voon

Date: June 2000

## ACKNOWLEDGEMENTS

The author wishes to express his sincere gratitude and appreciation to his project supervisors Professor M.S.J. Hashmi of Dublin City University and Dr. A.M.S. Hamouda of Univeristi Putra Malaysia for their valuable advice, guidance and willingness to share their expertise throughout the course of this study. The suggestions and advice had indeed contributed a lot and enhanced the author's scope of knowledge in this field.

The author would also like to express his appreciation to all the members of the school, especially Dr. M.A. El-baradie, Mr. B. MacDonald, Dr. P. Young, Dr. L. Looney, Ms M. Considine, Ms M. Reddy and Mr. L. Domican for their help and encouragement throughout these years.

The author wishes to acknowledge the contributions of all my fellow students in Dublin City University for their support and friendship. Special thanks to Dr. M.Y. Ismail, Dr. J. Hashim, Dr. J.C. Tan, J. Stokes, N. Khamkham, M. Hemry, K. Bakkar, X.P. Sun, B. O'Sullivan, P.H. Lim, R. Dumitrescu, D. Chuah, R. Padmanabhan, B. Hossain, H. El'Sheikh, Dr. S. Chowdhury, Dr. M. Iqbal, E. Mukhtar and Dr. G. O'Donnell

Special appreciation goes to the author's senior, colleague and housemate, S.H. Tang, for his friendship and advice.

The author wishes to express his special thanks and appreciation to his grandmother, who passed away, parents and sibling for their supports and good cheer. Sincere thanks to the author's relatives for their unconditional supports in both financial and spiritual to the author and his family during the hard time.

Last but not least, the author must gratefully remember the patient support and encouragement from his wife, W.H. Lau, throughout this long process.

# Development of Explicit Finite Difference-Based Simulation System for Impact Studies

SHAW VOON WONG, B.ENG.

## ABSTRACT

Development of numerical method-based simulation systems is presented. Two types of system development are shown. The former system is developed using conventional structured programming technique. The system incorporates a classical FD hydrocode. The latter system incorporates object-oriented design concept and numerous novel elements are included.

Two explicit finite difference models for large deformation of several material characteristics are developed. The models are capable of handling impulsive and constant load, impact and gravitational force, etc. Their capabilities of handling relatively complex shapes, which are not possible to deal with using classical FD models, are shown and discussed. A robust algorithm is suggested to describe general stress-strain relationship, ranging from elastic-perfectly plastic to multiple elastic modulii and plastic modulii with or without strain rate sensitivity. Hysteresis and Bauschinger effects are included as well.

Development of contact-impact treatments is presented. Three novel contact elements are designed, developed and validated. Node-node and node-element contacts are formulated with all the elements developed. The contact-impact treatments include a novel method in handling potential voids and overlaps at contact intersection. Rigid Coulomb's friction model is incorporated to handle sliding conditions.

Formulation of a novel revolute joint element is illustrated and several possible methods are suggested for handling bending limits. Incorporation of resistive moment and damping effects are presented. The numerical stability of the revolute element is discussed and a stability criterion is proposed.

The latter simulation system is used to model an idealised human body. Information from literatures, such as height and weight ratio of human segments, and resistive moment of human joints, is incorporated into the human model. Simulations are carried out according to a $2^k$ factorial design of experiment to find the main effect(s) contributed to human head/brain injury when subjected to frontal collision. A simple motorcycle model based on Malaysian KRISS 110 motorcycle is developed as well. The human model is successfully integrated into the motorcycle model. Possibility of solving such model with the developed simulation system is studied.

# TABLE OF CONTENTS

# List of Figures

# LIST OF TABLES

# LIST OF CODING

# Chapter 1: Introduction & Background

## 1.1 Importance and Justification of the Development and the Study

The computational simulation has become important component in research world. This is because computational simulations may be carried out instead of full-scale experiments. The advancement of computing in both hardware and software architectures significantly reduced the cost of running such simulations, especially comparing to the cost of conducting the experiments. Furthermore, it gives high flexibility in varying the system parameters, which is usually difficult or costly to achieve with the real experimental set-up.

Finite difference (FD) method in solving numerical simulation has been well known for decades. A great number of finite difference models for handling different problems have been developed and successfully implemented for the past 30 years. Another alternative is the finite element (FE) method. The main difference between finite element and finite difference methods is the data handling techniques for solving dynamics problems. The use of the finite element method has increased significantly in recent years. The success of the finite element method is believed to be due to the standardised data-handling organisation, which enables better implementation when involving new formulation of material, structure or other constraints. Furthermore, the success in developing powerful and general-purpose numerical solvers makes the FE method more popular. A new way of implementing the FD solver is required. Most of the FD and the FE solvers were written in FORTRAN in the 70s and the beginning of 80s. The advantage of Object-Oriented Programming (OOP) technique has not been taken in the formulation foundations, as the concept did not exist at that time. The disadvantage of the general FD models can be solved with the help of OOP technique.

Witmer *et al.* [1] developed a general FD element, which can handle one-dimensional structure subjected to large deformation. The model was chosen in this work initially as the preliminary general 1-D model for the FD system. Different implementation amendments of the coding have been made to solve various impact dynamic problems. The initial generalised FD model by Witmer *et al.* has limitation in implementation, which hardly acts as a component in a general FD solver. Complications occur when trying to solve structure branching problems. A new generalised FD model is required. Furthermore, supportive components, such as pre/post-processor, and supportive elements, such as contact element, are still a large area to explore. More efficient methodologies and formulations are still in demand.

Simulation in predicting human body motion subjected to collisions or crashes is important from safety in design aspect. Live experiment is beyond the reach of the researchers. Only coincident accident clips captured can be used for the study. This information is always limited and has very low repeatability. The use of human dummies is very popular in crash test, from full scale to sled test. But, the dummy itself is a dead object, and it does not represent the real human body. The simulation is more cost and time effective where the requirement of frequent change the crash and human parameters is vital towards the study of the crash and human motion mechanisms. Each full-scale crash test requires not less than 20,000 pound Sterling and each sled test requires not less than 2,000 pound Sterling at an established research centre.

## 1.2 OVERVIEW OF FINITE ELEMENT (FE) METHOD

FE method was first introduced in the 1950s. It has been continually developed and improved since then. With the help of today's computing technology, the method has been proving to be an extremely sophisticated tool for solving numerous engineering problems. It is widely used and accepted in many branches of industry. Developments in FE method have not been paralleled by any other numerical analysis procedure, and it has made many other numerical analysis techniques and experimental testing methods redundant. FE method is a straightforward and logical procedure following a well-defined path.

In the first step in dealing with the application of FE method, a body of interest (the problem) is divided into an assembly of subdivisions called elements, which are considered to be interconnected at joints, known as nodes. The variable is assumed to act over each element in a predefined manner, with the number and type of elements chosen so that the variable distribution through the whole body is adequately approximated by the combined elemental representations. This process is called discretisation. The formulations of element properties, stress or thermal for example, may be found in a variety of ways. The formulations share a constant format. Then the calculation of the equations for each occurrence of individual element is carried out. The nodal co-ordinates, material properties and loading conditions of the element are substituted into the general format. The equations for individual element are then assembled to obtain the global (system) equations, which describe the behaviour of the body as a whole. A general form is shown as the following:

$$[k]\{\mathbf{U}\} = \{\mathbf{F}\} \tag{1-1}$$

where $[k]$ is a square matrix, known as the stiffness matrix, and, $\{\mathbf{U}\}$ and $\{\mathbf{F}\}$ are the nodal displacements vector and applied nodal forces vector respectively. The solution of Equation (1-1) is not trivial in practice because the number of equations involved tends to be very large [2]. The matrix $[k]$ is normally very huge and holds more than 100x100 matrix elements for a very simple problem. Consequently $[k]$ cannot be inverted easily. After obtaining the global matrices, several methods can be used to solve the unknown variables. The most straightforward approach is the minimisation of the potential energy of the system. The approach is known more generally as variational formulation. This method is particularly useful to solve static problems (please refer to section 1.4: Static, Quasi-static and Dynamic Problems).

In a dynamic analysis, the effects of inertia forces are included in the calculations. These inertia forces are proportional to the acceleration of the body under investigation, and consequently introduce a time variation into the system equations. Solution of the equations yields some form of time varying or dynamic response of the body. The basic equations for the dynamic behaviour of a structure or body are expressed as the following in general:

$$[M]\{\ddot{\mathbf{U}}\}+[C]\{\dot{\mathbf{U}}\}+[k]\{\mathbf{U}\}=\{\mathbf{F}_t\} \tag{1-2}$$

where $[M]$ and $[C]$ are the mass matrix of the structure or body and corresponding damping matrix respectively. The mass matrix is formulated by summing the individual element mass matrices. The mass integration of an element can be lumped at the nodes (for explicit temporal integration scheme) or consistent (for implicit temporal integration scheme). $\{\ddot{\mathbf{U}}\}$ and $\{\dot{\mathbf{U}}\}$ are the nodal accelerations and velocities respectively. $\{\mathbf{F}\}$ is the summing forces vector. Different temporal integration schemes may be applied to obtain the transient response of the structure or body. The temporal integration schemes can be divided into explicit integration scheme and implicit integration scheme.

## 1.3 OVERVIEW OF FINITE DIFFERENCE (FD) METHOD

Discretisation is required for the FD method as well. In general a body or component is discretised into finite sections (may be referred as elements). Furthermore, the time domain is discretised as well by placing a grid on the temporal axis with grid spacing $\Delta t$ when dealing with dynamic problems. Then approximated "properties" are placed on these grids. The approximation can be partial difference equations. The FD method is very similar to the FE method in dealing with a dynamic problem, although they are not exactly the same. Anyhow, both methods need discretisation, element definition (element formulation in FE and grid approximation in FD). For a dynamic problem, Equation (1-2) is used also in FD method. Either the explicit temporal integration or the implicit temporal integration is applied.

A pseudo code [3] is shown in Figure 1-1. It is the basic algorithm to solve a time-dependent dynamic problem, which may consist of partial differential equation. In the initialisation subroutine, parameters are set, "uold" (the value of respective variables at the previous instant of time) is initialised, boundary conditions are defined. "unew" is the value of respective variables at next time step.

This basic implementation scheme is the same as the FE method. The main difference between them is the way to store information. The FD method does not suggest any specific procedure to store the data and resulting information. The FE method requires a systematic way to store all the information in matrix form.

```
Call Initial
For time <= Final time
    Call solution scheme
    If desired, Call output
    Set uold=unew
next time
```

Figure 1-1: General FD implementation pseudo code

*A difference scheme $L_k^n u_k^n = G_k^n$ approximating the partial differential equation $Lv = F$ is a pointwise convergent scheme if for any x and t, as $(k\Delta x, (n+1)\Delta t)$ converges to $(x,t)$ as $\Delta x$ and $\Delta t$ converge to 0. … by Thomas [3].*

## 1.4 STATIC, QUASI-STATIC AND DYNAMIC PROBLEMS

Numerical problems involving structure in general are divided into three types, which are static, quasi-static and dynamic. For a static problem the final responses of the system (the problem) will rest at a stable position, whether deformed or not deformed. A static problem has equilibrium due to the external and internal forces acting on the system.

$$\sum \mathbf{F}_i = 0 \tag{1-3}$$

where $\mathbf{F}_i$ is the external nodal force vector of node *i*. Example of a static problem would be a cantilever beam subjected to a constant distributed load and the maximum deflection is of interest. For a quasi-static problem, static solution is necessary, but a small amount of dynamics in the solution is acceptable. Metal forming simulation is a good example of quasi-static problem. In a quasi-static scenario, the equilibrium is as follows:

$$\sum \mathbf{F}_i \approx 0 \tag{1-4}$$

For a dynamic problem, the solution will not rest at a particular stage and can be defined if the following equation is observed.

$$\sum \mathbf{F}_i = m\mathbf{a} \tag{1-5}$$

where $m$ and $\mathbf{a}$ are the total mass and effective acceleration of the system. All impact and crash problems are under this type.

## 1.5 INTRODUCTION TO TEMPORAL INTEGRATION SCHEMES

Dynamic problems may be solved in two fundamentally different ways, either integration over the time domain or solution in the frequency domain [2]. In the first of these, the forcing function is divided into a number of impulses, which can be integrated over time. In the second, the force function is decomposed into its frequency components and the solution is found in the frequency domain. The latter approach is not widely used. Two different methods are available in time domain solution, namely direct integration and model superposition. Direct integration is the most general method and can be used for both linear and non-linear problems. On the other hand, model superposition is only suitable for linear system. For the interest of this section, only direct integration of time domain solution is discussed. Several approaches can be considered in direct integration method, and called as temporal integration schemes.

A temporal integration scheme is required to integrate the equations of motions. The scheme defines the relations among displacement, velocity, acceleration and time. Temporal schemes can be categorised into two main groups. They are explicit integration scheme and implicit integration scheme.

The main difference between the explicit and implicit schemes is their application/response prediction nature. The explicit scheme predicts time history responses whereas the implicit scheme gives most stable static deformation. Generally, the implicit scheme is incapable of solving dynamic problems, while the explicit scheme is not suitable in predicting static problems. The explicit scheme is able to solve static problem but with an oscillatory solution. A considerable damping is required to bring the structure from vibration to rest.

Three well-known integration schemes are illustrated in this section. They are two implicit schemes, namely Newmark scheme and Houbolt scheme, and one explicit scheme, namely central difference method.

## 1.5.1 NEWMARK'S SCHEME

The Newmark's scheme is an implicit scheme, where two user-defined parameters are used to control the stability and accuracy characteristics of Newmark's algorithm. They are represented with symbols $\beta$ and $\gamma$. Some commercial implicit solvers use Newmark's scheme with $\beta$ = ¼ and $\gamma = 0$. The use of Newmark temporal integration scheme was shown by literature [4] to produce undesirable oscillations in solving some contact problems. The formulation of the Newmark's scheme is based on the assumption that the average acceleration between time steps is constant. The following equations are the scheme's fundamental formulations.

$$\overline{\mathbf{u}}_n = \mathbf{u}_n + \Delta t\dot{\mathbf{u}}_n + \frac{\Delta t^2}{2}(1-2\beta)\ddot{\mathbf{u}}_n \tag{1-6}$$

$$\dot{\mathbf{u}}_{n+1} = \dot{\mathbf{u}}_n + (1-\gamma)\Delta t\ddot{\mathbf{u}}_n + \gamma t\ddot{\mathbf{u}}_{n+1} \tag{1-7}$$

$$\ddot{\mathbf{u}}_{n+1} = \frac{1}{\beta\Delta t^2}(\mathbf{u}_{n+1} - \overline{\mathbf{u}}_n) \tag{1-8}$$

where $\mathbf{u}_n$, $\dot{\mathbf{u}}_n$ and $\ddot{\mathbf{u}}_n$ are the displacement, velocity and acceleration vectors respectively at time step *n*. This method is unconditionally stable.

## 1.5.2 HOUBOLT'S SCHEME

Houbolt method is a three-step, second order accurate technique. For linear problems it is unconditionally stable. It requires a special starting procedure. Newmarks's method can be used as the starting algorithm for the first three steps. The formulations of Houbolt's scheme are as below:

$$\dot{\mathbf{u}}_{n+1} = \frac{11\mathbf{u}_{n+1} - 18\mathbf{u}_n + 9\mathbf{u}_{n-1} - 2\mathbf{u}_{n-2}}{6\Delta t} \tag{1-9}$$

$$\ddot{\mathbf{u}}_{n+1} = \frac{2\mathbf{u}_{n+1} - 5\mathbf{u}_n + 4\mathbf{u}_{n-1} - \mathbf{u}_{n-2}}{6\Delta t^2} \tag{1-10}$$

The above velocity and acceleration equations do not require velocities and accelerations from the previous Newton-Raphson iterations.

## 1.5.3 CENTRAL DIFFERENCE SCHEME

Central difference scheme is an explicit integration scheme. It is second-order accurate and conditionally stable. The scheme could be obtained from Newmark's scheme by setting $\beta = 0$ and $\gamma = ½$. The governing semi-discrete equations can then be solved to produce time history response. The formulations are given as the following:

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \dot{\mathbf{u}}_{n+0.5}\Delta t \tag{1-11}$$

$$\dot{\mathbf{u}}_{n+0.5} = \dot{\mathbf{u}}_{n-0.5} + \ddot{\mathbf{u}}_n\Delta t \tag{1-12}$$

$$\ddot{\mathbf{u}}_n = \mathbf{M}^{-1}\mathbf{F}_n \tag{1-13}$$

where $\mathbf{M}^{-1}$ and $\mathbf{F}_n$ are mass matrix and the summed force vector (which consists of external force, internal force, etc), at time step *n* respectively. Linear change in displacement in between time steps is assumed in the formulation. LS-DYNA3D[*] uses central difference method as its temporal integration scheme.

## 1.6 EXPLICIT VERSUS IMPLICIT

The main difference in terms of the user concern about explicit procedure and implicit procedure is their yielded-solution. The former shows transient response throughout the interested time frame. While the latter produces the solution as a whole, regardless of the time frame. Some basic differences in terms of procedure are given in the following tables.

Table 1-1: Basic difference in explicit procedure and implicit procedure

| Explicit | Implicit |
|---|---|
| • No matrix inversion | • Matrix inversion required |
| • Very small time steps | • Large time steps |
| • Stability limit | • Unconditionally stable |
| • No convergence check required | • Requires convergence checks |
| • Lumped mass | • Consistent or Lumped mass |

Table 1-2: Factors influencing computational efforts in explicit procedure and implicit procedure

| Explicit | Implicit |
|---|---|
| • Model size | • Model size |
| • Element edge length | • Degree of nonlinearity |
| • Young's modulus | • Number of time steps |
| • Density and mass | |
| • Termination time | |

[*] LS-DYNA3D is developed under Livermore Software Technology Corporation.

Table 1-3: User efforts in explicit procedure and implicit procedure

| Explicit | Implicit |
|---|---|
| • Fine tune the model for good and fast solution<br>• Check the results to validate kinetic energy | • Find convergence criteria for the solution<br>• Check for convergence after solving, repeat until solution is done |

Table 1-2 and Table 1-3 show the factors, which influence the computational efforts and the user efforts required for both explicit and implicit procedures. In terms of application, explicit solution can solve non-linear dynamic problems very efficiently. Static problems can be solved as well, but further implementation has to be done. Whereas, implicit solver can hardly solve large deformation (highly non-linear) dynamic problems, but is most suitable to handle static problem. When handling contact-impact problem, explicit solver is much better suited. Figure 1-2 shows the comparison of suitability of explicit and implicit solvers in solving different problems.

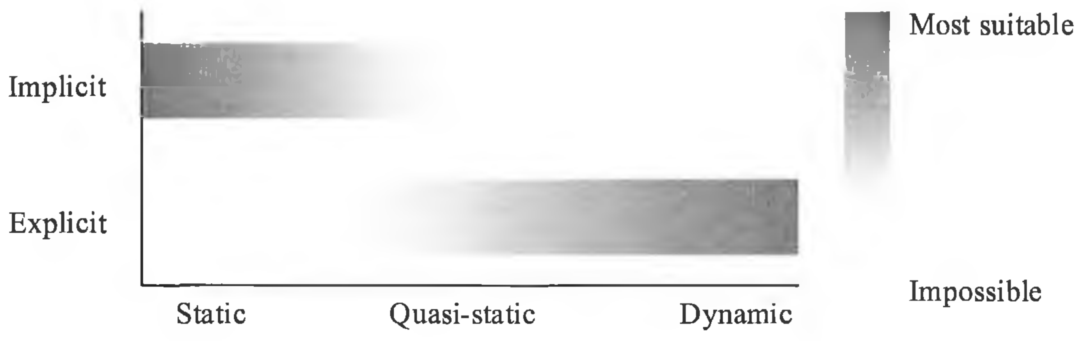


Figure 1-2: Suitability of explicit and implicit applications

If an exact solution is necessary, the user can achieve it by adding system damping in an explicit solver. To do so, the lowest natural frequency of the structure needs to be known. For the loading, a cubic equation for the loading curve instead of a sudden load (impact) should be used. The duration of the loading curve or ramp should be slightly greater than the natural period of the structure. Figure 1-3 shows a typical loading curve mentioned above. Under critical mass weighted damping is applied to eliminate any oscillation. For simplicity purpose, one can simply apply damping to the system to avoid any oscillation. Sacrifice of accuracy is expected. An approximation method is available to be used, which averages the maximum and minimum values during the oscillation and the final stage of the solution is treated as the static solution.

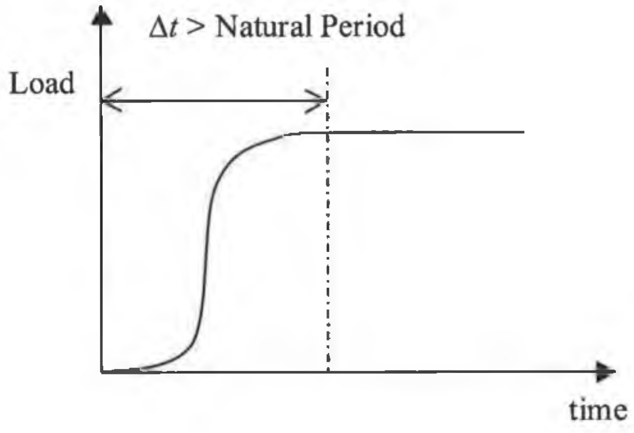


Figure 1-3: Typical loading pattern

As a whole, when dealing with dynamic problems, especially with dynamic contact, non-linear geometry, non-linear material, and with large deformation, explicit solver should be used. If static solution is required, then implicit solver is the better choice.

## 1.7 STANDARD COMPONENTS OF A NUMERICAL SIMULATION SYSTEM

A numerical simulation system normally consists of three main components. They are called pre-processor, processor and post-processor. One can guess the functions of these components from their respective names.

Pre-processor is used to develop the model with minimum amount of input from the user. The main role of the pre-processor is to create and define elements and nodes, which includes material model and loading. There are essentially two methods in which the element mesh can be generated: direct user input and automatic mesh generation. In the first of these, one defines each and every node and element. Thus, the whole mesh is built manually. For the latter method, the mesh is generated automatically by the compute (with some sort of meshing algorithms). Initially a CAD drawing is developed to represent the problem, and then a mesh is generated with selection of meshing choices. After the meshing, one has to define the element type of each element (may be defined in a group) in the meshed model. Initial loading and boundary conditions are to be defined then. A processor input data file is created at the final stage.

The input data file is then read into the processor. Different approaches, be it FD method or FE method, are implemented to retrieve the input data and solve the problem. Raw output data file(s) is generated which is to be processed or interpreted further in the Post-processor.

The post-processor produces equally impressive and convincing graphical output. The processed data from the processor is further interpreted in the post-processor to produce required information, such as deformation history, etc. Furthermore, the solution from the explicit solver needs to be validated with the kinetic energy, whereas convergence is required for implicit solution.

The development of the pre- and post-processors does not only make the user remote from the method, but also gives the programs the level of accuracy. All simulation only gives accurate and reliable results when used correctly. It is important to remember that both FD and FE simulation are approximated methods, and the validity and accuracy of the model and its solution rely on an accurate representation of the problem and correct analysis procedures.

## 1.8 COMMERCIAL PACKAGES

Several commercially available packages for impact studies are presented in this section. Some characteristics are shown and some information about the developers is provided.

### A. HEMP3D

HEMP3D was developed by Wilkins *et al.* [5] of Lawrence Livermore National Laboratories in 1975. The equations of motion is a second order accurate finite difference formulation. The initial code in 2D was presented by Wilkins in 1964 [6]. Constitutive relations are given as below:

$$\dot{s}_{ij} = \begin{cases} 2G\left[\dot{\varepsilon}_{ij} - \dfrac{\dot{V}}{3V}\right], & i = j \\ G\dot{\varepsilon}_{ij}, & i \neq j \end{cases} \tag{1-14}$$

$$\dot{\varepsilon}_{ij} = \begin{cases} u_{i,i}, & i = j \\ \dot{u}_{i,j} + \dot{u}_{j,i}, & i \neq j \end{cases} \tag{1-15}$$

where $G$ is the shear modulus, $\varepsilon_{ij}$ is the strain components, $\dot{s}_{ij}$ is the stress deviators, and $V$ is the volume and $\dot{V}/V = u_{i,j}$ from continuity. These equations are for isotropic materials.

Intersecting sliding surfaces are permitted with separation and interaction being dependent on forces acting on the interface. The sliding surface logic was not shown in any accompanied document. A semiautomatic-zoning program, named VMESH3D, acts as its pre-processor. The pre-processor allows users to define blocks and automatically subdivides these into finer zones. The blocks may then be assembled to form the final mesh. HEMP3D has a post-processor called VPIX3D. It is used to verify the input data and interpret output. The VPIX3D is capable of generating 3D picture in line drawing and halftone image with shading keyed to a selected flow-field variable.

## B. EPIC-3

EPIC-3 was developed by Johnson [7] of Honeywell inc. EPIC-3 is an FE solver. Tetrahedral element is used. Tetrahedral and triangular elements offer inherently greater resistance to distortion than cubic or quadrilateral elements. Artificial stiffness and some symmetries are introduced into calculations, which is not desirable. EPIC-3 applies similar basic equations to HEMP3D. Strain-rate and temperature effects are taken into account with the following equations

$$\sigma_D = [\sigma(\varepsilon) + c_1 \ln(\bar{\dot{\varepsilon}})][c_2 + c_3 T] \tag{1-16}$$

Where $\sigma_D$ is the dynamic stress, $\sigma(\varepsilon)$ is a strain-dependent effective stress, $\bar{\dot{\varepsilon}}$ is the equivalent strain rate, $T$ is the temperature, $c_1$, $c_2$ and $c_3$ are user-supplied constants. The temperature is defined as:

$$T = T_0 + \frac{E_s}{c_s \rho_0} \tag{1-17}$$

Where $T_0$ is the initial temperature in an element, $E_s$ is the internal energy per unit original volume, $c_s$ is the specific heat and $\rho_0$ is the initial density.

An extensive slide-line logic is used, which permit sliding with or without friction, void formation and collapse, tunnelling and erosion of material. Intersecting sliding planes can be treated with the current algorithm. Pre-processor of EPIC-3 allows rapid, automatic generation of rectangular and circular plates, hollow or solid uniform or tapered rods, hollow or solid spheres and hollow or solid hemispherical, ogival and conical nose shapes. EPIC-3 has an extensive post-processor, which variety of geometry plots can be performed. In addition, contour plots of pressure, normal and shear stresses, equivalent to plastic stress and strain, temperature, initial energy and plastic work are readily obtained. Time-history plots may also be obtained for all energies, angular momentum and velocities, system centre of gravity positions, linear momentum and velocities, nodal positions, velocities and accelerations, element pressures, stresses and temperatures.

## C. DYNA3D

DYNA3D was first developed by Hallquist [8] of Lawrence Livermore National Laboratories in 1976. Lately, the author under Lawrence Software Technology Corporation developed LS-DYNA3D. The previous version employs both the HEMP difference equations and a Popov-Galerkin finite element formulation using four-, six-, or eight-node quadrilateral solid elements. The present version contains only finite element option. A lot of elements such as beams, shells, rigid bodies, single surface contact, interface friction, discrete springs and dampers, optional hourglass treatments, optional exact volume integration. It contains more than 40 material types at the latest version, which includes rubber and foam. Automated contact input for all input types, automatic single surface without element orientation and constraint technique for contact can be applied. LS-DYNA3D is seeing extensive use by many of the world's top automobile, aerospace, and ordnance companies, by government and corporate laboratories and research organisations, and in universities.

Finite Element Model Builder (FEMB) is the pre- and post-processor of LS-DYNA3D. It comes free with LS-DYNA3D as a package. FEMB acts as the pre- and post-processor at the same time. The user only requires familiarising with single environment. Some other commercial available pre-post-processors support LS-DYNA3D as well. FEMB has an impressive list of pre-processing functions enabling engineers to produce high quality FEA models. It has an automatic surface-meshing algorithm, which easily eliminates a majority of the time required to mesh trimmed and standard IGES surfaces. One main drawback from its pre-processing is lacking of user-friendliness comparing to other pre-processors of its class. As a post-processor, FEMB can quickly post-process results including real time animation of stresses, strain energy, displacements and time history curves. Animation adds clarity and realism to the analysis results when post-processing. FEMB can take the process of displaying stress contour and deformed shape results one step further with real time animation by providing the most accurate representation of the analysis results.

## D. HULL

HULL was first written in 1971 as a purely hydrodynamic code and rewritten in 1976 to permit computations involving elastic-plastic flow by Matsuka and Durrett [7] of Orlando Technology inc. In 1979, additional development began to link HULL with the 3D Lagrangian EPIC-3 code. HULL employs a second-order accurate finite difference scheme. The following equations are used to define the constitutive relations.

$$\dot{s}_{ij} = 2G\left(\dot{\varepsilon}_{ij} - \delta_{ij}\dot{u}_{k,k}/3\right) \tag{1-18}$$

$$\dot{\varepsilon}_{ij} = \left(\dot{u}_{i,j} + \dot{u}_{j,i}\right)/2 \tag{1-19}$$

where $\delta_{ij}$ is the Kronecker delta.

A rezoner is included that permits a continuous translation of the computational mesh at a continuous velocity in any one or all of the co-ordinate directions and is implemented in 3D. A mesh generator namely KEEL is used as HULL's pre-processor. KEEL defines the initial conditions, mesh co-ordinates, and material property specifications for 3 components of velocity, density, internal energy, mass, and volume of each computational cell. Material properties for over 35 solids, liquids and gases can be generated automatically by specification of a material number. The mesh generator permits construction of a fine computational mesh in the region of most interest and gradual increase of zone sizes away from that region. KELL includes a number of geometric configurations for insertion of materials, tracer particles and stations for collection of flow-filed data. PULL is named as its post-processor. It is capable to generate up to 20 contours of a flow field variable (density, pressure, energy, etc) versus a spatial co-ordinate plot. Furthermore plots of vector quantities indicating direction and magnitude can be produced.

The above mentioned features and comments are based on the author's knowledge and literature available. Newer versions of the above packages may have new features, which are not mentioned in this thesis.

## 1.9 DEVELOPMENT ALTERNATIVES

Several of computer programming languages can be used to develop the system. FORTRAN[*] was a famous programming language in dealing with scientific and mathematics formulation, which is very suitable for developing such a system. This is the main reason why a lot of the available packages were developed with FORTRAN. Besides the other conventional languages, like BASIC and Pascal, new generation computer languages should be considered. Some of them are Visual BASIC and Visual C++. C++ is chosen owing to its speed and performance, which overwhelms the rest of the old and new generation languages. Furthermore, it supports state of art programming techniques, which is proven crucial in developing a big system. One drawback is the difficulty to learn and use for development especially dealing with user-interface development in Windows[†] environment.

## 1.10 C PROGRAMMING LANGUAGE

The development of Visual C++ is based on the current C++ standard. The conventional C (ANSI[‡] C) acts as a foundation to the development of the C++.

C language is developed by Dennis Ritchie who was a programmer in AT&T[§] Bell Laboratory in 1972. It has a very close relationship with UNIX operating system since it was created. The UNIX operating system was developed by using C language. As the result, C language is the main programming language used by the UNIX users. At the initial stage, C language and UNIX were used only for PDP-11, a minicomputer owned by DEC[**]. UNIX has become the major operating system ranging from microcomputer,

[*] FORTRAN stands for FORmula TRANslator. An old computer programming language.

[†] Referring to both Microsoft Windows 9x and Windows NT platforms.

[‡] American National Standard Institution

[§] American Telegraph & Telephone

[**] Digital Equipment Corporation, an American Corporation

RISC* workstation, minicomputer, mainframe to CRAY, and the supercomputer. At the same time, C language has gained its reputation and becoming the most welcoming and most popular programming language.

C language is classified neither a high level (such as COBOL) nor a low level programming language (such as Assembly Language). C has been classified in between the two levels. C has the capability of structure programming which is essential for a high level programming language. This feature cannot be found in a low level programming language. C is also able to perform low level operation and low level programming which is very close to assembly language and machine language. A high level programming language does not have this feature normally. This feature enables C to be used to generate almost all computer process and fully utilise capability of the microprocessor of a computer.

Nowadays, structure programming alone is not enough to overcome the needs of the new era. With introduction of object oriented programming, C++ was developed. The Visual C++ is the latest generation of the C family by Microsoft.

### 1.10.1 MICROSOFT VISUAL C++

The Microsoft compiler package provides a comprehensive, up-to-date production-level development environment for developing all Windows applications. Visual C++, version 4.0, provides a powerful 32-bit compiler to help develop Win32 applications for Windows 9x and Windows NT operating system. Visual C++, version 2, provides a 16-bit compiler for developing 16-bit Win32s applications for Windows 3.x and MS-DOS environments.

The latest Visual C++ compilers incorporate many new and upgraded features. Some of the most important enhancements include support for the AT&T C++ 2.1 standard, precompiled headers, auto-in-lining and p-code (packed code).

* Reduce Instruction Set Computer

The Microsoft Visual C++ compiler package also provides tools for building Windows programs for Windows 3.x, Windows 9x and Windows NT environments. The program code can even be leveraged to hardware platforms like Apple Macintosh and other RISC machines. The C++ compiler includes all the header files, libraries, dialog and resource editors necessary to create a truly robust Windows application. Microsoft has also incorporated the resource editors for bitmaps, icons, cursors, menus, and dialog boxes directly into the integrated environment. And speaking of integration, new class Wizards help to build Object Linking and Embedding (OLE) applications using the Microsoft Foundation Class (MFC) libraries in record time.

### 1.10.2 FEATURES OF VISUAL C++

Visual C++ includes share and reuse components for faster application development by leveraging pre-existing components [9]:

- Component Gallery adds new features through pre-built objects including OLE controls.
- OLE controls add functionality to user applications. Controls can be built and to be shared with others, even those using Visual Basic, Visual FoxPro, and other tools.
- Latest MFC gives user prewritten classes and a solid foundation to build on.
- Customisable AppWizard speeds development of company- or industry-specific applications.
- Integrated source-code control lets user share and leverage code across teams.

The development environment of Visual C++ simplifies C++ programming:

- *ClassView* lets user go beyond files as you work with your projects as a set of objects instead of files.
- *WizardBar* lets user navigate quickly to your objects with minimum effort.
- Improved usability through enhancements to the text editor (including BRIEF and Epsilon emulation), the debugger, the resource editors, and customisation.

- Developer Studio integrates Visual Test, Fortran PowerStation, Microsoft Development Library, and Visual SourceSafe to ensure seamless access to all your tools.
- *Books online* places over 15,000 pages of information at user fingertips.

Visual C+ includes an innovative build system to speed the edit-build-debug cycle for large projects:

- Incremental compiler compiles only the functions that have changed.
- Incremental linker re-links only modules that have changed.
- Minimal rebuild builds only the files that need to be built.

## 1.11 OBJECT-ORIENTED PROGRAMMING (OOP)

As the matter of fact, object-oriented programming is not a new programming concept. Scott Guthery states that "object-oriented programming has been around since subroutines were invented in the 1940s" [9]. The article continues by suggesting that objects, the foundation of object-oriented programming, have appeared in earlier languages, such as FORTRAN II.

Considering these statements, why object-oriented programming is only heard in the closing decade of the 1900s? Why is object-oriented programming being touted as the newest programming technique of the century? It seems that the bottom line is packaging. OOP concepts may have been available in 1940, but they were not packaged in a usable container.

Early programmers, growing up with the BASIC language, often wrote large programs without the use of structured programming concepts. Huge volume of programming code was tied together with one- or two-letter variables that had a global scope. Goto statement abounded. The code was a nightmare to read, understand, and debug. Adding new features to such a program was like unlocking Pandora's box. The code, to say the least, was very difficult to maintain.

In the 1960, structured programming concepts were introduced suggesting the use of meaningful variable names, global and local variable scope, and a procedure-oriented top-down programming approach. Applying these concepts made code easier to read, understand, and debug. Program maintenance was improved because the program could now be studied and altered one procedure at a time. Programming languages such as Ada, C and Pascal encouraged a structured approach to programming problems.

Bjarne Stroustrup is considered the father of C++ and developed the language at Bell Labs in the early 1980s. He may well be the father of object-oriented programming as it was only introduced in the C++ language. Jeff Duntemann pronounced that "Object-oriented programming is "structured of structured programming". It is the second derivative of software development, the Grand Unifying Theory of program structure" [9].

Object-oriented Programs function differently from the traditional procedural approach. They require a new programming strategy that is often difficult for traditional procedure-oriented programmers to grasp. An object-oriented program is a program that consists of a group of objects that are often related. With C++, an object is formed by using the new class data type. A class provides a set of values (data) and the operations (methods or member functions) that act on those values. Developers can then manipulate the resulting objects by using messages. In OOP, objects hold not only the data (member data) but the methods (member functions) for working on that data. The two items have been combined into one working concept. Simply put, objects contain data and the methods for working on that data.

There are three distinct advantages offered to the programmer by OOP. The first is program maintenance. Programs are easier to read and understand, and object-oriented programming controls program complexity by allowing only the necessary details to be viewed by the programmer. The second advantage is program alternation (adding or deleting features). One can often make additions and deletions to programs, such as in a database program, by simply adding or deleting objects. New objects can inherit everything from a parent object, and they only need to add or delete items that differ. The third advantage is that a programmer can use objects numerous times. One can save well-designed objects in a toolkit of useful routines that can be easily inserted into new code, with few or no changes to that code.

## 1.12 MICROSOFT WINDOWS 95/NT 4.0 PROGRAMMING

If one has programmed for Windows in C, one knows that the word class was used to describe the definition of a window long before C++ programming came to Windows. A window class is vital to any Windows C program. A standard structure holds the data that describes this window class, and a number of standard window classes are provided by the operating system. A programmer usually builds a new window class for each program and registers it by calling an API* function, RegisterClass(). Windows that appear on the screen can then be created, based on that class, by calling another API function, CreateWindow().

### 1.12.1 A C-STYLE WINDOWS CLASS

The WNDCLASS structure, which describes the window class, is equivalent to the WNDCLASSA structure. WINUSER.H sets up two very similar window class structures, WNDCLASSA for programs that use normal strings, and WNDCLASSW for Unicode† programs. WINUSER.H is code supplied with Developer Studio. It's typically in the folder \Program Files\DevStudio\VC\include. If one were creating a Windows program in C, one would need to fill a WNDCLASS structure [10].

### 1.12.2 HANDLES

A handle is more than just a pointer. Windows programs refer to resources like windows, icons, cursors, and so on with a handle. Behind the scenes there is a handle

* API stands for Application Programming Interface

† Unicode is a two-byte character set standard enabling programs to work with international languages whose alphabet requires more than 8-bits of storage.

table that tracks the address of the resource as well as information about the resource type. It's called a handle because a program uses it as a way to "get hold of" a resource. Handles are typically passed around to functions that need to use resources, and returned from functions that allocate resources.

There are a number of basic types of handles: HWND for a window handle, HICON for an icon handle, and so on. No matter what kind of handle is being used, remember it's a way to reach a resource so that you can use it.

### 1.12.3 ENCAPSULATING THE WINDOWS APPLICATION PROGRAMMING INTERFACE (API)

Microsoft Windows was designed long before the C++ language became popular. Because thousands of applications use the C-Language Windows API, that interface will be maintained for the foreseeable future. This guarantees that C++ applications will be able to coexist with C applications [11].

API functions create and manipulate windows on the screen, handle drawing, connect programs to Help files, facilitate threading, manage memory, and much more. When these functions are encapsulated into Microsoft Foundation Classes (MFC), the programs can accomplish the same basic Windows tasks, with less work on the developer's part.

### 1.12.4 MESSAGES AND COMMANDS

If there is one thing that sets Windows programming apart from other kinds of programming, it is messages. Most DOS programs, for example, relied on watching (sometimes called polling) possible sources of input like the keyboard or the mouse to await input from them. A program that wasn't polling the mouse would not react to mouse input. In contrast, everything that happens in a Windows program is mediated by messages. A message is a way for the operating system to tell an application that something has happened, for example, the user has typed, clicked, or moved the mouse, or the printer has become available. A window (and every screen element is a window) can also send a message to another window, and typically most windows react to messages by passing a slightly different message along to another window.

A command is a special type of message. Windows generates a command whenever a user chooses a menu item, clicks a button, or otherwise tells the system to do something. In older versions of Windows, both menu choices and button clicks generated a WM_COMMAND message; whereas, nowadays WM_COMMAND for a menu choice and a WM_NOTIFY for a control notification like button clicking or list box selecting are available. Commands and notifications get passed around by the operating system just like any other message, until they get into the top of OnWndMsg(). At that point, Windows message passing stops and MFC command routing starts.

Command messages all have, as their first parameter, the resource ID of the menu item that was chosen or the button that was clicked. These resource IDs are assigned according to a standard pattern—for example, the menu item File, Save has the resource ID ID_FILE_SAVE.

Command routing is the mechanism OnWndMsg() used to send the command (or notification) to objects that can't receive messages. Only objects that inherit from CWnd can receive messages, but all objects that inherit from CCmdTarget, including CWnd and CDocument, can receive commands and notifications. That means a class that inherits from CDocument can have a message map. There won't be any entries in it for messages, only for commands and notifications, but it's still called a message map.

## 1.13 MICROSOFT FOUNDATION CLASS (MFC)

Windows applications are easy to use; however, they are not as easy to develop. Many programmers get waylaid by having to master the use of hundreds of Windows API functions required to write Windows applications.

Microsoft's solution to this steep learning curve is the object-oriented Foundation Class Library. The reusable C++ classes are much easier to master and use. The MFC library takes full advantage of the data abstraction offered by C++, and its use simplifies Windows programming. Beginning programmers can use the classes in a "cookbook" fashion, and experienced C++ programmers can extend the classes or integrate them into their own class hierarchy. MFC library represents virtually every Windows API feature and includes sophisticated code that streamlines message processing, diagnostics, and other details that are a normal part of all Windows application. This logical combination and enhancement of Windows API functions has nine key advantages [12]:

- *The Encapsulation of Windows API is Logical and Complete.* The MFC library provides support for all of the frequently used Windows API functions, including windowing functions, messages, controls, menus, dialog boxes, graphics device interface (GDI) objects (fonts, brushes, pens, and bitmaps), object linking, and the multiple document interface (MDI).

- *The MFC Functions are Easy to Learn.* Microsoft has made a concerted effort to keep the names of the MFC functions and associated parameters as similar as possible to their Windows API parent classes. This minimises the confusion for experienced Windows programmers wanting to take advantage of the simplified MFC platform. It also makes it very easy for a beginning Windows programmer to grow into the superset of Windows API functions when they are ready or when the application requires it.

- *The C++ Code is More Efficient.* An application will consume only a little extra RAM when using the classes in the MFC library compiled under the small memory model. The execution speed of an MFC application is almost identical to that of the same application written in C using the standard Windows API.

- *The MFC Library Offers Automatic Message Handling.* The MFC library eliminates one frequent source of programming errors, the Windows API message loop. The MFC classes are designed to automatically handle every one of the Windows messages. Instead of using the standard switch-case statements, each Window message is mapped directly to a member function, which takes the appropriate action.

- *The MFC Library Allows Self-Diagnostics.* Incorporated into the MFC library is the ability to perform self-diagnostics. This means that you can dump information about various objects to a file and validate an object's member variables, all in an easily understood format.
- *The MFC Library Incorporates a Robust Architecture.* Anticipating the much-needed ANSI C throw or catch standard, the MFC library already incorporates an extensive exception-handling architecture. This allows an MFC object to eloquently recover from standard error conditions such as "out of memory" errors, invalid option selection, and file or resource loading problems. Every component of the architecture is upward compatible with the proposed ANSI C recommendations.
- *The MFC Library Offers Dynamic Object Typing.* This extremely powerful feature delays the typing of a dynamically allocated object until run time. This allows you to manipulate an object without having to worry about its underlying data type. Because information about the object type is returned at run time, the programmer is freed from one additional level of detail.
- *The MFC Library can Harmoniously Co-Exist with C-based Windows Application.* The most important feature of the MFC library is its ability to co-exist with C-based Windows applications that use Windows API. Programmers can use a combination of MFC classes and Windows API calls within the same program. This allows an MFC application to easily evolve into true C++ object-oriented code as experience or demand requires. This transparent environment is possible because of the common naming conventions between the two architectures. This means that MFC headers, types, and global definitions do not conflict with Windows API names. Transparent memory management is another key component to this successful relationship.
- *The MFC Library Can be Used with MS-DOS.* The MFC library was designed specifically for developing Windows applications. However, many of the classes provide frequently needed objects used for file I/O and string manipulation. For this reason, both Windows and MS-DOS developers can use these general-purpose classes.

- *The MFC Library and Wizards.* The ClassWizard and ControlWizard only create code compatible with the MFC. These dynamic program developers are a must when developing OLE applications.

## 1.14 HUNGARIAN NOTATION

Microsoft programmers use a variable naming convention called Hungarian Notation. It is so named because it was popularised at Microsoft by a Hungarian programmer named Charles Simonyi, and probably because at first glance, the variable names seem to be written in another language.

In Hungarian Notation, the variable is given a descriptive name, like Count or ClassName, that starts with a capital letter. If it is a multi-word name, each word is capitalised. Then, before the descriptive name, letters are added to indicate the type of the variable— for example, nCount for an integer, or bFlag for a Boolean (True or False) variable. In this way the programmer should never forget a variable type, or do something foolish like passing a signed variable to a function that is expecting an unsigned value.

Table 1-4: Common Hungarian Notations[10]

| Prefix | Variable Type | Comment |
|---|---|---|
| a | Array | |
| b | Boolean | |
| d | Double | |
| h | Handle | |
| i | Integer | "index into" |
| l | Long | |
| lp | Long pointer to | |
| lpfn | Long pointer to function | |
| m_ | Member variable | |
| n | Integer | "number of" |
| p | Pointer to | |
| s | String | |
| sz | Zero terminated string | |
| u | Unsigned integer | |
| C | Class | |

The style has gained widespread popularity, though some people hate it. The structures used by the API and the classes defined in MFC all use Hungarian Notation. The author tended to follow the notation during the development of the simulation system. But, not the whole of the Hungarian Standard Notation is followed. Some of the Hungarian prefixes are as Table 1-4.

## 1.15 MULTITASKING WITH WINDOWS THREADS

When using Windows 95 (and other modern operating systems), several programs can be executed simultaneously. This ability is called multitasking. Many of today's operating systems also allow threads, which are separate processes that are not complete applications. A thread is a lot like a subprogram. An application can create several threads (several different flows of execution) and run them concurrently. Threads give you the ability to have multitasking inside multitasking. The user knows that he or she can run several applications at a time. The programmer knows that each application can run several threads at a time.

Using multiple threads can lead to some interesting problems. For example, how to prevent two threads from accessing the same data at the same time? What if, for example, one thread is in the middle of trying to update a data set when another thread tries to read that data? The second thread will almost certainly read corrupted data, since only some of the data set will have been updated.

Trying to keep threads working together properly is called thread synchronisation. Event objects can be used for synchronisation purpose. Besides that, critical sections, mutexes, and semaphores can make thread programming safer.

Critical sections are an easy way to ensure that only one thread at a time can access a data set. When you use a critical section, you give your threads an object that they have to share between them. Whichever thread possesses the critical-section object has access to the guarded data. Other threads have to wait until the first thread releases the critical section, after which another thread can grab the critical section in order to access the data in turn. Because the guarded data is represented by a single critical-section object, and because only one thread can own the critical section at any given time, the guarded data can never be accessed by more than a single thread at a time.

Mutexes are a lot like critical sections but are a little more complicated, because they enable safe sharing of resources not only between threads in the same application, but also between threads of different applications.

Although semaphores are used like critical sections and mutexes in an MFC program, they serve a slightly different function. Rather than allowing only one thread to access a resource at a time, semaphores allow multiple threads to access a resource, but only to a point. That is, semaphores allow a maximum number of threads to access a resource simultaneously. When you create the semaphore, you tell it how many threads should be allowed simultaneous access to the resource. Then, each time a thread grabs the resource, the semaphore decrements its internal counter. When the counter reaches 0, no further threads are allowed access to the guarded resource until another thread releases the resource, which increments the semaphore's counter.

Multithreaded-program seems simple to write, but it is NOT. In reality, it needs a very high "skill". A lot of things have be decided at the initial stage, such as what to be processed as the child thread, what to be placed in the critical section, what to share among which threats, when and how the data is to be updated, etc. Furthermore, a single mistake will cause the developer a huge effort to debug. As it is hard to know which thread has really causing the interruption. Thus, for a normal program, multithreads should be avoided to reduce complications. It does not worth the attempt.

## 1.16 AIMS OF STUDY

The Objective of the present study can be summarised as following:

1. Develop a numerical method based simulation system for impact studies
2. Employ and implement an existing general purpose FD hydrocode [1] (which found not suitable later)
3. Design, develop, implement and validate new general FD hydrocode to cope with all possible configurations
4. Design, develop, implement flexible material model
5. Design, develop, implement and validate supportive elements such as contact-impact element and revolute element
6. Preliminary simulation study on human motion subjected to frontal crash.
7. Preliminary simulation study on human motion subjected to frontal crash with motorcycle.
8. Conceptual initial design of the sled test rig (towards building a test rig for validation).

## 1.17 METHOD OF APPROACH

The progress of the development and the study of the subject followed the following flow chart.

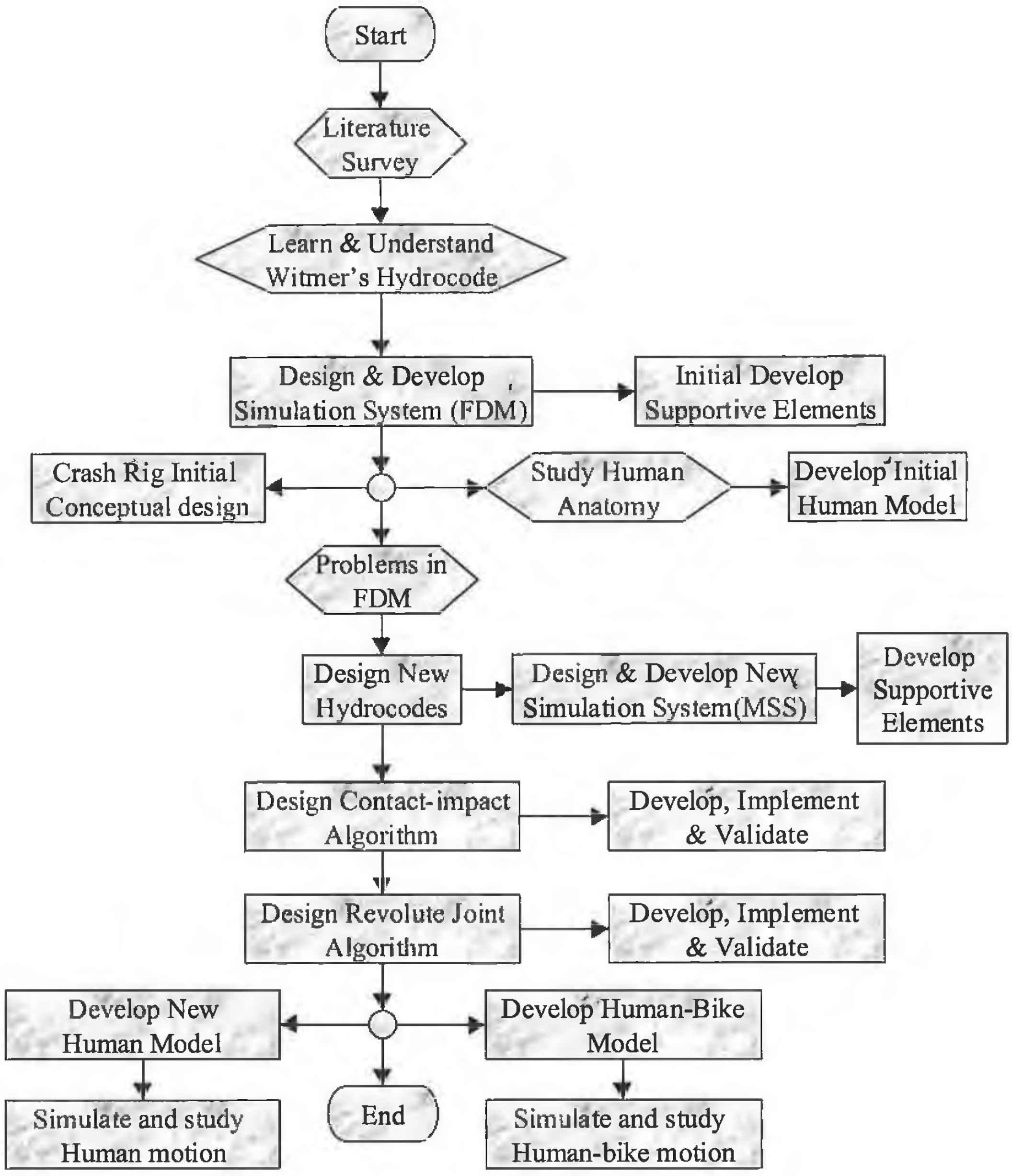

Start
Literature Survey
Learn & Understand Witmer's Hydrocode
Design & Develop Simulation System (FDM)
Initial Develop Supportive Elements
Crash Rig Initial Conceptual design
Study Human Anatomy
Develop Initial Human Model
Problems in FDM
Design New Hydrocodes
Design & Develop New Simulation System(MSS)
Develop Supportive Elements
Design Contact-impact Algorithm
Develop, Implement & Validate
Design Revolute Joint Algorithm
Develop, Implement & Validate
Develop New Human Model
Develop Human-Bike Model
Simulate and study Human motion
End
Simulate and study Human-bike motion

Figure 1-4: Project Progress and Approach

## 1.18 LAYOUT OF THESIS

This thesis is divided into seven chapters. Following this introductory chapter, chapter two gives a critical review of the relevant literature, from development of hydrocodes, contact algorithms to human motion simulation. Development of the first generation of simulation system is revealed in chapter three. The object-oriented second-generation simulation system is presented in the following chapter. Chapter five illustrates the development and the formulation of elements used, which include the constitutive equations. Validations and numerical examples of respective elements are also shown. The simulation system is used to simulate human motion during frontal impact. This study is presented in chapter six. Derivation of human body parameters and development of human body model is explained. In addition, incorporation of motorcycle in the study is carried out. Preliminary study of the human body motion is conducted. Finally in chapter seven, conclusions are drawn and recommendations for future work are suggested based on the present work.

# Chapter 2: LITERATURE SURVEY

## 2.1 INTRODUCTION

Description of previous works is provided in this chapter. The chapter presents the literature survey done in the area of hydrocodes, numerical FD and FE models, multibody simulation systems, contact-impact algorithms, revolute joint element, human body anatomy, human mathematical models, human joints modelling, motorcycle crash, and human injuries in motorcycle crash.

## 2.2 HYDROCODES

A hydrocode may be loosely defined as a code for solving large deformation, finite strain transient problems that occur in a short time scale [13]. In contrast to structural static analysis codes, the energy equation is integrated in time and the deviatoric and pressure terms in the stress tensor are usually modelled separately. The solution is advanced in time using an explicit scheme. According to Benson [13], explicit finite element and finite difference based hydrocodes are rarely discussed in detail in textbooks.

## 2.3 FINITE DIFFERENCE AND FINITE ELEMENT MODELS FOR IMPACT STUDIES

Benson [13] pointed out in his paper that derivation of some elements are not unique from either finite element (FE) or finite difference (FD) viewpoint. Also noted that virtually all finite difference and finite element programs used seems to be the same. Furthermore, there are very few algorithms that cannot be derived from both the finite element and finite difference viewpoints. One advantage the finite element method has over the finite difference method is a natural means for ensuring that the forces are always work conjugate to the displacement. The equivalence between the finite element and finite difference formulations is first shown in reference [14].

Finite difference method is one of the most commonly used numerical methods. Simplicity is gained through the nature of the formulation. Conventionally, finite difference method is good in representing the system but more problem-oriented. Generalised finite difference method is a demanding subject. Different explicit finite difference models have been developed for the past 30 years to solve impact problems. Leech *et al.* demonstrated an 1-D explicit finite difference model in predicting the dynamic response of shells subjected to dynamic loads in reference [15] which is one of the first models to handle impact problems.

Explicit scheme is an incremental formulation. The numerical models can be divided into two main different approaches. They are called Eulerian (some are called moving co-ordinate or updated formulation) and Lagrangian respectively [16]. Bathe *et al.* classified Eulerian formulation where all static and kinematic variables are referred to an updated configuration in each load step. All static and kinematic variables are referred to the initial configuration in Lagrangian formulation [16]. Classical coding of Eulerian formulation can be found in references [1, 17-21] and Lagrangian formulation in references [22-24]. Comparison between Eulerian and Lagrangian formulations has been done by Yamada [25], which shows a maximum difference of 25 percent in the displacement when predicting a simple truss structure. Comparison of these two methods were made by Stricklin *et al.* [26] and Dupuis *et al.* [27]. Witmer *et al.* [1] suggested a general-purpose explicit finite difference model to predict the response of a 1-D structure moving in a 2-D space, which involved large deformation and subjected to impulsive loading. Hashmi *et al.* [28] suggested a 1-D explicit finite difference method in predicting the dynamic responses of certain structures subjected to impact and impulsive load in reference. Noh suggested an integral difference technique in reference [29] and has been implemented in HEMP [30] and PISCES [31]. Some other finite difference coding each has its own finite difference formulation, such as SHALE [32], SALE[33], TENSOR[34] and YAQUI [35].

Specific elements were being developed to handle some special configurations or setting. Vasudaven *et al.* [36] developed a new space curved frame finite element to be used for crash analysis with the intention to handle each member of a crash problem with a single element. The development was based on Iura and Atluri's beam model [37]. Vasudaven *et al.*'s latest model employed a mixed variational principle and using rotations as independent variables [38]. Hughes *et al.* developed an effective thick plate bending finite element model with reduced integration techniques [39]. Mistakidis *et al.* [40] proposed a new method which incorporated FE models to solve examples involving connected structures.

Woodward [41] demonstrated two versions of 1-D finite difference model moving in 1-D space, considering different factors in predicting penetration of thick metal targets by projectiles. Mushrooming phenomenon was described by a simpler formulation suggested by Hashmi *et al.* [42]. Comparisons to experimental results have been made in both literatures. Meanwhile, Abbas *et al.* [43] used finite difference method in predicting forces exerted upon a rigid structure by an impacting hypothetical projectile. The impact force was calculated on the basis of conservation of mass, momentum and energy.

Similar dynamic hydrocodes can be applied to study fracture mechanics and stress pulse. Chen [44] used a Lagrangian finite difference formulation in HEMP to study the dynamic stress intensity factors. Later, the problem was repeated in references [45-47] with FE method and boundary element method. Lately, Lin and Ballmann [48] amended Chen's FD method. Zhang and Ballmann [49] used an explicit finite difference scheme for elasto-dynamic problems to analyse contact impact of crack edge. A FD model based on integration along bi-characteristics [50] was used to predict the stress wave propagation in elastic and elastic/viscoplastic solids, in reference [51] and [52] respectively. Liu and Tanimura [53] studied dynamic response, tensile stress concentration and distribution of maximum tensile stress in a rectangular block subjected to impact using the FD model mentioned above. Kaizu *et al.* [54] implemented Clifton's model [50] to study the dynamic behaviour of thick plates due to oblique impact. Meanwhile, Huang and Wang [55] used Lagrangian FD dynamic code to model the fracture process in a ductile solid, where plate impact tests are performed for modelling the dynamic damage and fracture process.

### 2.3.1 RECENT DEVELOPMENTS

Very recently, new computational technologies have been incorporated into the development of numerical method based solver. Davalos and Rubinsky [56] suggested an evolutionary-genetic based FD formulation to analyse heat transfer problems. Lately, Rubinsky and Davalos [57] further implemented the evolutionary-genetic analogy in solving a simple two-dimensional physics problem using both FD and FE method. Parallel computational algorithm is getting its attention in numerical method based simulation system. Agrawal *et al.* [58] presented a super-element model based parallel algorithm for a plannar vehicle dynamics. The authors showed the feasibility of real-time simulation and control of multibody system using parallel computing.

Kothe *et al.* developed a data-parallel algorithm, which was used in PAGOSA [59], a three-dimensional Eulerian finite difference wave propagation code. Kimsey and Olson presented a parallel algorithm and data structures for implementing multimaterial, two-step Eulerian finite difference solution schemes on hypercube architectures in reference [60]. A data-based approach was suggested by Gandhi and Hu [61], where analytical models are derived directly from crash test measurements using system identification techniques. Recently, an Object-oriented library was developed by Lage [62] to support the formulation of boundary elements. The members of the several important classes were shown and panel clustering method [63] was implemented.

## 2.4 WITMER *ET AL.*'S MODEL AND ITS EXTENSIONS

Witmer *et al.* [1] and Hashmi *et al.* [28] demonstrated the feasibility of using the respective finite difference models in simulating response of general simple structures impact under loading. Hashmi and Thompson [42] amended the FD model to simulate the deformation of impacting flat-ended projectiles, which is called the mushrooming problem. Later, Woodward and Lambert [64] enhanced the model by considering the constant flow stress, which predict better responses depending on geometry.

Witmer *et al.*'s model was then modified for simulating several specific problems [65-68]. Argyris *et al.* suggested an extension version of Witmer *et al.*'s model in formulating a shell element for crash analysis in reference [69-71]. Edge folding of a U-section beam was simulated with the model and with market available finite element solver, TRUMP, for comparison purpose. Mesh density study was shown in the paper. Two bending models were incorporated. Those modes are compared from the point of view of an actual crash analysis of the front end of a car [72]. The edge folding of a bending model has a decisive advantage over another model. This concerns the definition of the shell geometry in the course of the bending deformation. Argyris *et al.* continued the work in the formulation of the shell element. Satisfactory applicability of the model for engineering purpose was demonstrated on an elasto-plastic shell segment exposed to a detonation loading in reference [73].

The modified models were specific and problem-oriented. The generality was not there as in the models of the first two. Witmer's model is a generalised finite difference model. It handles dynamic response of a simple shell structure and takes into account the effects of elastic-plastic or elastic-linear strain hardening behaviour, strain rate sensitivity and large deformation. Both formulations by Witmer *et al.* and Hashmi *et al.* models neglected rotary inertia and transverse shear deformation with infinite shear rigidity. The models are lack of flexibility in representing the structures and problems. General finite difference models for structural impact simulation described above are not suitable in use without the presence of the developers. The models are designed to represent a simple open-ended structure. Boundary conditions have to be clearly specified for different problem representations.

## 2.5 MULTIBODY SIMULATION SYSTEMS

Multibody system formulation is used widely to solve a broad variety of engineering problems such as aerospace, civil engineering, vehicle design, micromechanical analysis, robot motion and biomechanics. The dynamics of multibody systems is based on classical mechanics. The simplest multibody element is a free particle, which can be treated by Newton's equations [74]. Later, Euler [75] introduced the rigid body principle element. Free body principle (resulting in reaction forces) was proposed by Euler to handle constraints and joints. The well-known Newton-Euler equations were presented to handle multibody dynamics. In the 1980s, complete software systems for the modelling, simulation and animation were offered on the market as described by Schwertassek and Roberson [76].

In general, the elements used in multibody systems consist of rigid bodies and/or particles, bearings, joints and supports, springs and dampers, active force and/or position actuators [77]. Figure 2-1 shows typical components of a multibody system.

Otter *et al.* [78] have defined the unique and standard description of multibody elements to be used for all kinds of multibody system coding. Setting a standard on multibody system coding and formulation towards international standardisation is on going. The following are the two main characteristics defining a multibody system [77]:

- A multibody system consists of rigid bodies and ideal joints. A body may degenerate to a particle or to a body without inertia. The ideal joints include the rigid joint, the joint with completely given motion (rheonomic constraint) and the vanishing joint (free motion).
- The topology of the multibody system is arbitrary. Chains, trees and closed loops are admitted.

Computer-aided analysis of crashworthiness and structural impact has received considerable attention and application of multibody formulation is popular nowadays. Rigidity of multibody systems is not a problem at present. Multibody dynamic formulations were being used to solve structural dynamic problems. Structural and vehicle crashworthiness can be approximated by a multibody constrained mechanical system where revolute joints and non-linear revolute springs representing plastic hinges. This kind of model has been successfully used in vehicle crash simulations [79, 80, 81]. Cardona and Geradin [82] describe a finite element formulation of flexible multibody systems where the flexible members are treated in a fully non-linear manner using exact beam models.

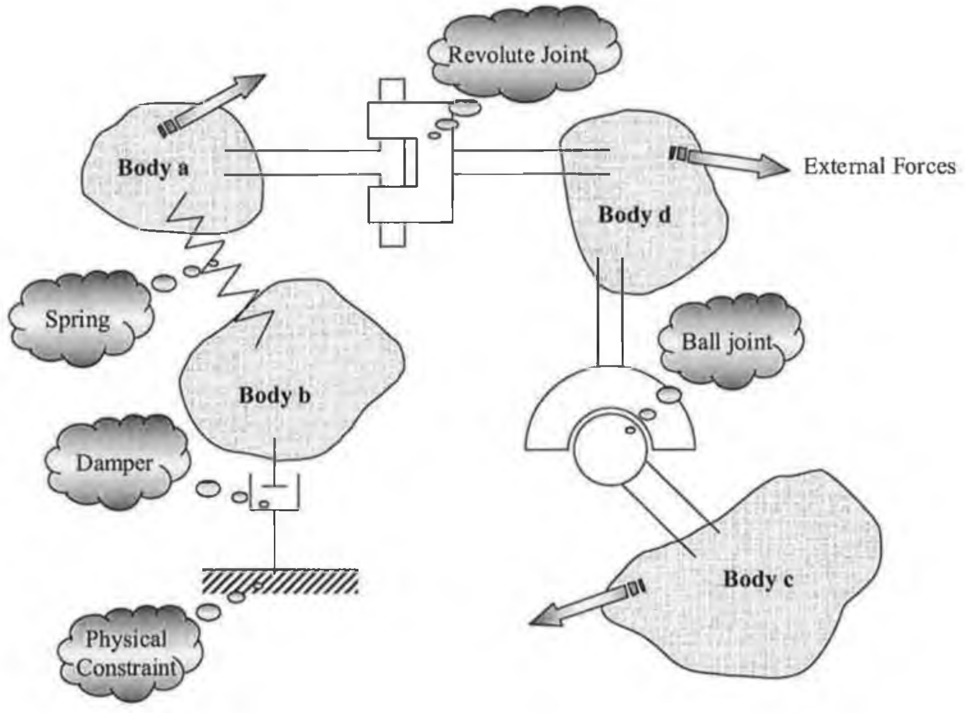


Figure 2-1: Schematic representation of typical multibody system components

Ambrosio [83] applied multibody dynamic formulation to study the structural systems experiencing geometric and material non-linear deformations. The components of the structure underwent large relative rotations and displacements. FE method is used to describe the flexible structure, which are able to undergo a large rigid body motion. Application to the crashworthy behaviour of a train coach is shown by Ambrosio's model. Earlier before Ambrosio, Yang and de Peter [84] used multibody methods to predict the response of a railway vehicle running on a tangent or a curved track. Lately Pereira *et al.* [85] presented a multibody dynamic formulation for systems with linear and non-linear structural deformations and plastic-hinge modelling approach was applied to a rigid-flexible multibody system is presented. In the paper, the approach was applied to the crashworthiness analysis and design studies of railway vehicles. Ma and Lankarani [86] suggested a general methodology for kineto-static analysis of multibody systems with flexible structures undergoing large motion and complicated structural deformation. Rigid multibody dynamic can predict the gross motions and displacements at the boundaries and FE method can be used to determine the corresponding loads and deformations of the entire structure. A typical rigid/flexible multibody system is shown below [87].

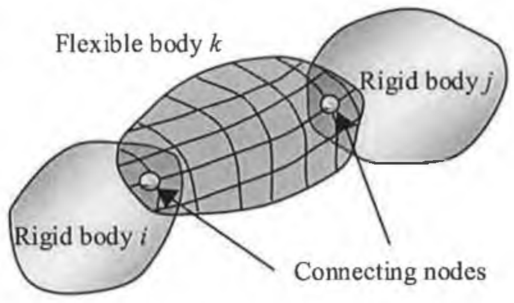


Figure 2-2: A flexible body connecting two rigid bodies

A Hybrid II anthropomorphic test dummy with FE model of the lumbar spine and multibody model as a whole can be developed as well, which are shown by references [86, 87].

Multibody solutions are dynamic transient in nature. Lee and Nikravesh presented a systematic methodology and formulation to determine the steady-state response of multibody systems in reference [88]. The equations of motion were in terms of the initial relative joint accelerations. A set of conditions was incorporated into the equations of motion to yield the steady-state equations. With this, steady-state response was predicted.

## 2.6 CONTACT-IMPACT ALGORITHMS

Contact-impact problem is one of the major components in any finite element (FE) or finite difference (FD) simulation system. Contact occurs while any two mechanical components or two parts of a single component start touching each other. Contact interactions may have a significant influence on the behaviour of the structure of the mechanical system.

## 2.6.1 CLASSIC CONTACT-IMPACT FORMULATIONS

Wallis presented a theory of collisions of either inelastic or elastic particles in 1670 (Wallis, *Mechanica, sive De Motu, Tractatus Geometricus*). Newton generalised the above collision results while discussing the third law of motion in relation to partly elastic collisions [89]. Classical collision law of Newton is one of the earliest models in tackling contact-impact problem. The following equation shows the basic relationship described by Newton's law of impact,

$$\mathbf{v}_\mathrm{f} \cdot \hat{\mathbf{n}} = -e\mathbf{v}_\mathrm{o} \cdot \hat{\mathbf{n}} \tag{2-1}$$

where $\mathbf{v}_\mathrm{f}$ is the rebound velocity after the impact and $\mathbf{v}_\mathrm{o}$ is the initial velocity before impact. The value $e$ is a coefficient called coefficient of restitution, which is to be determined through experiments and the value is between zero and one. The model is widely accepted and used nowadays.

The subject of contact mechanics may be said to have started in 1882 with the publication of H. Hertz's classical paper, *On the contact of elastic solids* [90]. Hertz found a contact law of a simple analytical form while two equal bodies compressed. The contact law is restricted to small relative velocities and progress of contact impact has been largely associated with the investigation of elastic-plastic stress waves. Also pointed by Hertz, the elastic waves during impact must be reflected many times in the bodies in order to assure quasi-static contact. It is true for longitudinal impact of thin rods, but not during the impact of a rigid sphere on infinite elastic half-space, where no waves are reflected at all. Subsequent findings [91, 92] show compressive waves do exist at the latter contact-impact scenario. Kilmister and Reeve [93] stated an impact theory via Poisson's hypothesis. Central and collinear collisions were separated into a compression phase with a normal component of impulse, $\overline{P}$, followed by a restitution phase with normal impulse, $e\overline{P}$. Thus, $\overline{P}(1+e)$ is the normal component of impulse that acts on each body during the collision and $\overline{P} \geq 0$.

### 2.6.2 CONTACT ALGORITHM COMPONENTS

A complete general contact-impact algorithm consists of the following main components:

- Contact-impact modelling – involves formulation of the contact-impact mechanics as general. It includes the handling of nodal and element information for systematical processing.
- Contact searching – formulations involved in searching the contacted location, nodes and elements. It is very much depending on the contact-impact modelling. Pseudo element mapping for example.
- Normal contact-impact response – provides the reaction from the contact-impact in the normal direction to the contact-impact surface, such as Lagrangian multiplier and penalty method.
- Tangential frictional response - provides sticking and sliding effects such as Coulomb friction model.

### 2.6.3 CONTACT-IMPACT MODELS

Generally, contact can be described as either static contact or dynamic contact. In a static contact phenomenon, the two bodies in contact are in static equilibrium, otherwise it is a dynamic contact. A dynamic contact is often much more complicated than a static one. Contact-impact is often used to stress the dynamic effects in contact phenomenon. As a general, mechanical problems involving contacts are inherently non-linear and involve unknown boundary conditions. Contact-impact phenomenon can be further divided into small displacement-rotation and large displacement-rotation [94]. Contact problems can be classified as counterformal problems or conformal problems accordingly as the dimensions of the contact region are small or large, compared to the local radii of curvature of the contacting surfaces [95]. Contact problems are called "Hertzian" if the contact surface is approximately planar and the bodies have undergone small straining in the neighbourhood of the contact surface. Hertzian problems are subdivided from counterformal problems. Hughes *et al.* [96] have presented a finite element method for a class of Hertzian contact-impact problems in reference. Newmark's temporal integration was used.

The contact algorithms nowadays have gone beyond the Hertzian's limit. Paul and Hashemi [95] showed a solution in solving frictionless counterformal contact stress problems. Paul and Hashemi also illustrated the first known solution at that time for non-Hertzian contact stresses at the interface of a steel wheel on a steel rail. Meanwhile, Hallquist [97] detailed the numerical procedure for handling sliding interfaces and impact in large deformation explicit finite element and finite difference. The procedure was designed to handle four nodal constant stress quadrilateral zones in the spatial discretisation. Chaudhary and Bathe [98] suggested a solution method for analysing contact between two or more 3-D bodies. A Lagrange multiplier technique was employed. Later, pinball algorithm with penalty method and Lagrange method is suggested by Belytschko and Neal [99]. Pinballs or spheres were embedded in the elements. The contact constraint was enforced on the spheres rather than the elements themselves. An explicit finite difference procedure was suggested to handle contact-impact problem at crack edges in reference [100]. The model presented was limited to crack edges special arrangement scenario.

## 2.6.4 CONTACT SEARCHING

Hallquist represented a well-known master-slave surface algorithm for contact searching in reference [101]. His idea was further enhanced and implemented into the well-known explicit finite element package, LS-DYNA3D [102]. The algorithm suffers some drawbacks, with the most obvious being the infeasibility of handling single surface contacts. Later Benson and Hallquist [103] used nested bucket sort algorithm to group nodes in the new single surface contact algorithms. To prevent undetected penetrations at intersections of mater surfaces, normal nodal vector projection was replaced by segment normal vector in LS-DYNA3D. ANSYS, another well-known package, defines contacts in term of element. A contact element consists of a node from the contact surface and boundary nodes from the target surface [104]. Pseudo element mapping is used to handle potential voids and overlaps at contact intersection. The treatment of contact interfaces involving beam elements is most difficult [105]. Contacts on beams can only occur on the physical boundary of the beam not on the reference line.

Automatic contact offers the benefits of significantly reduced model construction time and fewer opportunities for user error, but faces significant challenges in reliability and computational costs [106]. Whirley and Englemann [106] exhibited a new four-step automatic contact algorithm. The proposed method can handle automatic identification of adjacent and opposite surfaces in the global search phase, and the use of a smoothly varying surface normal which allows a consistent treatment of shell intersection and corner contact conditions without ad-hoc rules.

## 2.6.5 NORMAL RESPONSE

Contact problems are normally characterised by contact constraints applied on contacting boundaries. Lagrangian multiplier [107, 108] and penalty method [109, 110, 111] are among the most commonly used. The penalty method is used in the explicit programs DYNA2D and DYNA3D as well as in the implicit programs NIKE2D and NIKE3D[102]. It is known that the addition of the penalty method, in contrast to the Lagrange multiplier method, always decreases the stable time step. According to linearised stability analysis, the stable time step for the central difference method is not decreased by a Lagrange Multiplier method [112]. The number of degrees of freedom is not influenced/ increased while using Penalty method. In the classical Lagrangian Multiplier method, contact tractions are considered as additional degrees of freedom.

Combined Lagrangian multiplier and penalty function was suggested by Jiang and Roggers [113]. De la Fuente and Felippa [114] treated the penalty function as an ephemeral fictitious non-linear spring which was inserted on anticipation of contact. A collision model was developed by Kawachi *et al.* [115]. The model used for simulating simultaneous collision impulse with friction between rigid bodies. Perturbed Lagrangian Method was proposed and used by a number of researchers [116, 117,118, 119]. The method can be considered as a generalisation of the Lagrangian Multiplier method where an additional term involving the contact traction is added to the variational equations. The classical Lagrangian Multiplier method is then obtained as a limiting case, while the penalty method is recovered by solving for the contact traction and eliminating them from the equilibrium equations [120]. Augmented Lagrangian method was suggested [120, 121]. It is used in optimisation where a function of displacements and Lagrangian multipliers is constructed so that its minimum corresponds to the solution of the constrained problem. These conventional contact elements suffer from mathematical inconsistencies and they require user defined and arbitrary parameters [122]. The value of these parameters can drastically affect the results.

Different contact-impact numerical solutions have been suggested in order to avoid the above mentioned shortcomings. Taylor and Flanagan [123] proposed contact constraint algorithm. The penetration force was computed as a function of the penetration distance which tried to push back the contact node (or slave node) back to the constraint boundary. Impact and release conditions are imposed to ensure momentum conservation. Variational inequalities have been suggested in dealing with contact problems in reference [124]. Refaat and Meguid [125-127] used variational inequalities in solving small displacement static contact problems. Quadratic programming and Lagrangrian multiplier were used to avoid the use of user defined penalty parameters. All the above mentioned methods have their strong and weak points.

Kilmister and Reeve contact-impact model [93] is a feasible model to be used. The author cannot find any literature for the implementation of the model in numerical contact-impact analysis.

### 2.6.6 TANGENTIAL RESPONSE / FRICTION

Coulomb's friction model has been widely used in predicting the tangential frictional forces [105]. The classical law regards the friction coefficient as a constant. Non-classical friction laws were introduced to describe the micro-slip phenomenon in reality. Figure 2-3 shows classical and non-classical friction models.

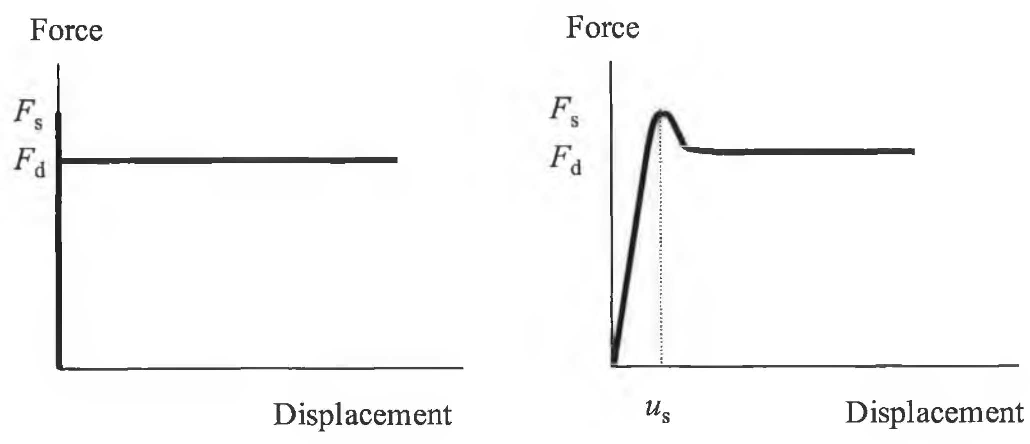


Figure 2-3: Frictional Models, (left: Classical; right: Non-Classical)

In these figures, $F_s$ and $F_d$ are the static frictional force and dynamic frictional force respectively. In the classical model, $F_s = \mu R$, where $\mu$ is the coefficient of friction and $R$ is the normal reaction force. Kilmister and Reeve [93] proposed a tangential impulse of $\mu\overline{P}(1+e)$ acting at the contact point if continuous slip occurred.

Barber [128] divided contact phenomenon into three regimes. They are gross slip, microslip and adhesion. Gross slip was defined where tangential relative motion (slip) occurs throughout the contact area. Microslip occurred where part of the contact area (usually a central circle) is in adhesive contact, whilst the remainder slips. No relative motion at any point in the contact area for the last phenomenon.

Non-linear friction law was proposed in reference [129] which the friction coefficient depends on the relative sliding of two contacting points. Fridricksson [130], Curnier [131] and Kikuchi and Oden [132] incorporated elasto-plasticity theory to formulate more general friction laws. The tangential frictional relations are used to define or describe the sticking and sliding conditions of a contact situation.

According to Zhong and Mackerle [105], the implementation of the classical friction law appears to be more difficult than that of the non-classical friction law introducing displacement dependence.

### 2.6.7 STABILITY AND SENSITIVITY OF THE TEMPORAL INTEGRATION SCHEMES IN CONTACT-IMPACT ALGORITHMS

Central difference method as temporal integration scheme is widely used in an explicit solver. Penalty method and Lagrangian multiplier are among the most commonly used to predict normal contact-impact response. It is known that the addition of the penalty method, in contrast to the Lagrange multiplier method, always decreases the stable time step. According to linearised stability analysis, the stable time step for the central difference method is not decreased by a lagrange multiplier method [99]. Karaoglan and Noor [133] assessed temporal integration schemes for the sensitivity analysis of frictional contact-impact response of axisymmetric composite structures. Central difference explicit method, Newmark implicit scheme and Houbolt implicit scheme were used for comparison. The following conclusions were drawn from the study:

- Explicit temporal integration schemes are more efficient compared to implicit schemes, while handling rigid surface contact-impact problems in conjunction with Lagrange Multiplier contact formulation

- The sensitivity analysis based on the penalty and perturbed Lagrangian formulation requires larger penalty parameters than those used in predicting the response.

- Accurate prediction of the sensitivity coefficients requires smaller time step increments than that used for predicting the response.

Sha *et al.* [134] proposed a new temporal integration scheme called forward incremental displacement – central difference method. The methodology was based on employing a variational inequality for dynamic problems involving coulomb friction.

Energy gains observed with some calculations of collision dynamics based on Newton's law when initial slip and friction exist. This unacceptable consequence of energy gains is only present when the collision is non-collinear [89]. Explanations were made by Keller [135] and Brach [136]. Slip reversal was blamed by the former. The latter illustrated that the phenomenon occurs most often when *e* is large and slip reverses in direction partway through the collision. Violation of the law of conservation of energy was reported in Ivanov's study [137] of the collision of two rigid bodies as well. Wang and Mason [138] suggested a new way to determine the resultant impulsive forces instead of the conventional Newton's way, where Poisson's hypothesis of restitution and Routh's [139] method were adopted.

## 2.7 REVOLUTE JOINT

Many machine components and devices consist of interconnected rigid bodies. Pin joint and revolute joint are among the common and the simplest connecting joints. In kinematics terms, pin and revolute joints have only one primary degree of freedom, which is a rotation about the pin axis. As a revolute joint in reality, rotational limits, rotational friction and viscous damping have to be considered. For a more complicated situation like human joint modelling, the effects of the muscles will contribute to the building of resistive moment at the respective joint.

The revolute joint limits are where the connected links/bodies cannot further rotate because of physical constraints in the nature of the revolute joint. In reality, the system may be designed to have rotational limits for some reasons. Furthermore joint locking mechanism may be required. One common example is the locking of individual solar arrays of a satellite during deployment. This operation results in a redistribution of total momentum that may lead to impulsive forces and moments on the system. Large vibration may occur especially involving light flexible structures. Damping is a crucial factor in many mechanical systems, including revolute joint and pin joint. The damping will be able to stable down post-limit oscillating excitation.

A great number of works has been carrying out in developing mathematical models and numerical simulation systems to study the structural flexibility of the members during their motion. Different approaches were employed. Among them are finite element method [140, 141], assumed mode method [142] and lumped parameter approach [143]. Several literatures [144, 145, 146] studied the post impact behaviour of closed loop mechanisms and a single flexible link. Momentum balanced was used but application of restitution coefficient was not involved. Locking and rotational limits were not studied. Not much work has been done in dealing with locking and/or rotational limits. Nagaraj *et al.* [147] studied the dynamics of a two-link flexible system with locking mechanism. Two revolute joints with torsion spring were employed. Mathematical models with momentum balanced were shown and comparisons to experimental results were made.

Computational formulation of revolute joint is rare to be found in literatures. Revolute or pin joint problems are normally characterised by physical constraints applied on nodes. To achieve an idealised fixed-point pin joint, simply apply linear displacement constraints and release one of the rotational displacement onto the coincide nodes of the connected bodies. Things become more complicated when the revolute or pin joint connecting two free bodies. The global constraints cannot be applied in this situation, but local constraints. Simple constraints (both local and global) application will no longer be capable of handling complex problems, which involve rotation limits, locking, resistive moment and damping.

A complete revolute element can be found in ANSYS, a well-known commercial finite element system. It is known as COMBIN7 [104]. The element consists of 2 primary nodes and 3 optional nodes. Figure 2-4 shows the main components of the ANSYS revolute element. Figure 2-5 shows the element structure.

The term *K* is the rotational spring stiffness when the element is locked or reaching its limit. The term *Tf* is the friction limit torque and the term *C* is the coefficient of rotational viscous damping. *θ*U and *θ*L are the upper (forward) rotation limit and the lower (reverse) rotation limit respectively. The figure shows clearly that the rotation of the connected bodies is constrained with a high stiffness rotational spring.

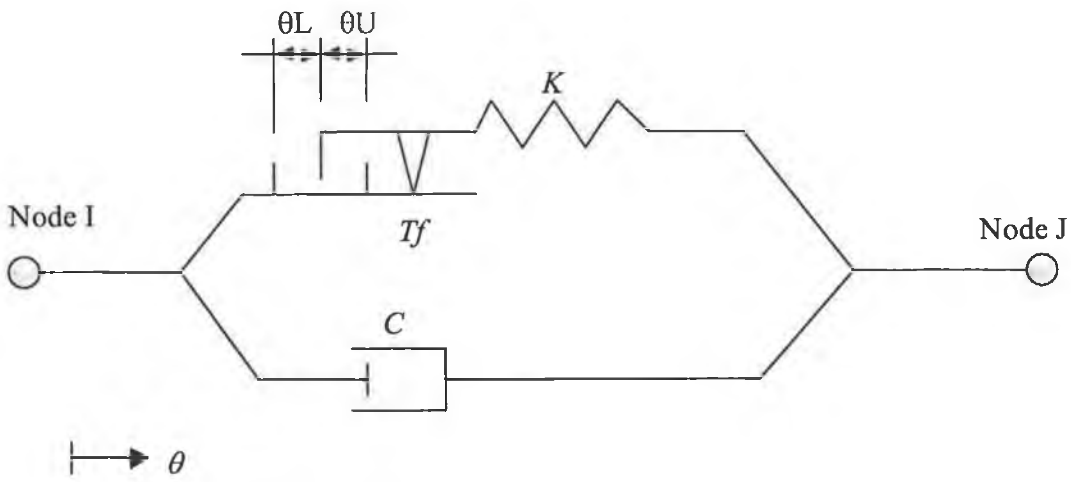


Figure 2-4: ANSYS COMBIN7 element dynamic behaviours [104]

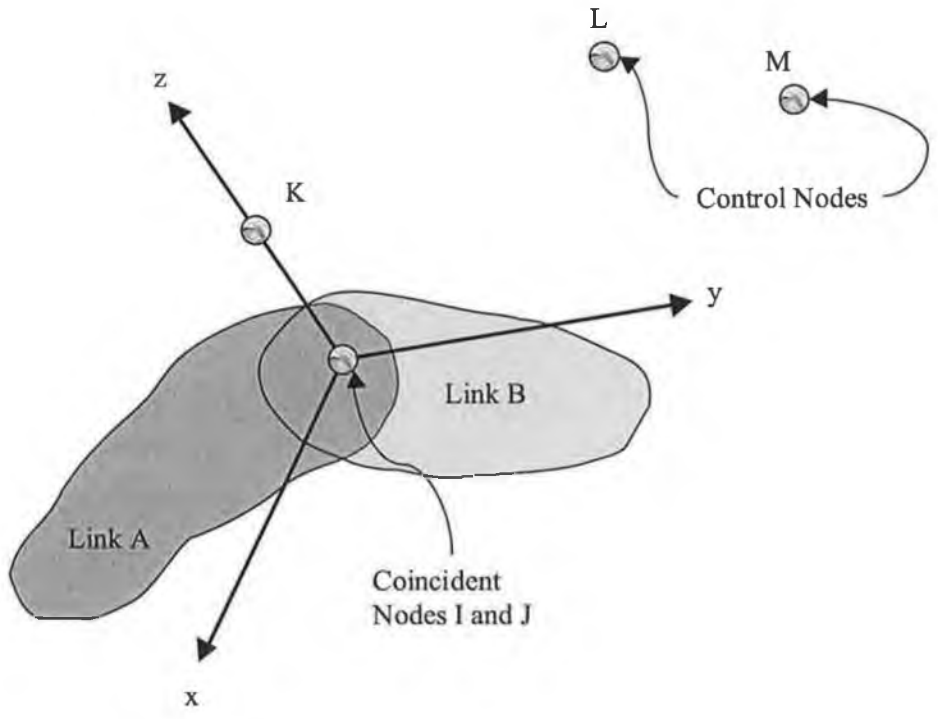


Figure 2-5: ANSYS COMBIN7 element structure [104]

The impact phenomenon is similar to the Penalty method of contact-impact element, where the normal contact force is depending on the contact stiffness. Thus, the revolute model suffers the same shortcoming with Penalty method, which a user predefined stiffness value is required. Low stiffness may cause significant rotational penetration even to penetrate through the constraint. High stiffness may cause over excitation, which gives unrealistic response. As a common practice, initial rotational spring stiffness after rotary limits is selected arbitrarily. Fine-tuning has to be carried out through trial and error to obtain a better result. Similar method is applied to handle torsion effect of connected bodies about the revolute joint, where the user defines springs' stiffness beforehand. The variable *Tf* defines the constant resistive rotation stiffness of the revolute element. Multiple rotation stiffness is prohibited with single revolute element. In addition, coexistence of the rotational stiffness and the rotational resistive moment is prohibited in the ANSYS element. Locking mechanism is not handled as well.

## 2.8 HUMAN BODY ANATOMY FOR HUMAN MOTION

The biomechanical analysis of a total body motion requires proper anatomical data on all body segments that may be classified as rigid. All body parts have soft tissue displacement that takes place especially during fast motions, such as impact/crash scenario, where high accelerations and decelerations occur. Soft tissue movement has not been dealt with successfully in the analysis of human motions, and so has been disregarded by biomechanists when calculating forces due to motion [148].

### 2.8.1 HUMAN BODY SEGMENTAL ANATOMY

The skeletal structure determines the segments to be considered rigid. The movement at the joints with masses of relatively unchanging length contribute to the rigidity. The main segments are the hands, forearms, upper arms, feet, shanks, thighs, head and neck, and trunk. The trunk may be subdivided into pelvis, abdomen and thorax. Figure 2-6 shows the location of each segment. The movement of each segment has to be analysed separately to obtain kinematics and kinetic data.

The human body moves by rotating each segment about one of its ends or about its long axis. In majority of the total body motion analysis, the body is treated as sixteen link pendular system. Soft tissue motion and motion of the extremities about their long axes are normally ignored. Critical anatomical data required for each segment are weight, length, centre of gravity and radius of gyration from each end [148].

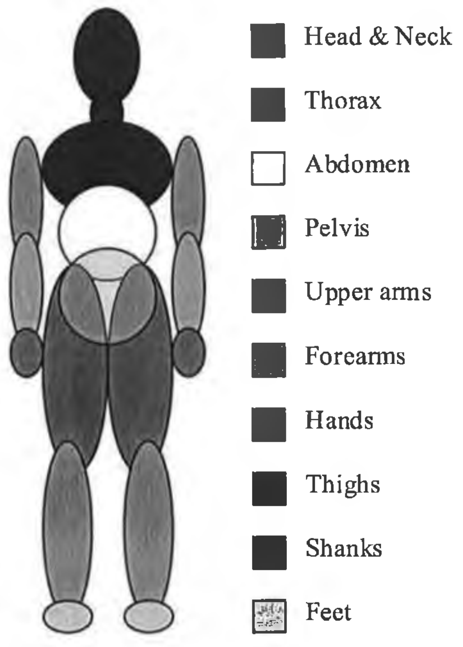


Figure 2-6: Human Segment Anatomy

### 2.8.2 VERTEBRAL COLUMN

The vertebral column is an important supporting structure of the human body. A normal vertebral column consists of 33 vertebrae. There are seven cervical, twelve thoracic, five lumbar, five scaral and four coccygeal. The first 24 separated by fibrocartilaginous intervertebral discs and sacrum and coccyx are fused together. Variations in vertebrae, which include number, position and shape, usually congenital and occur occasionally. Figure 2-7 shows the lateral view of a typical vertebral column [149].

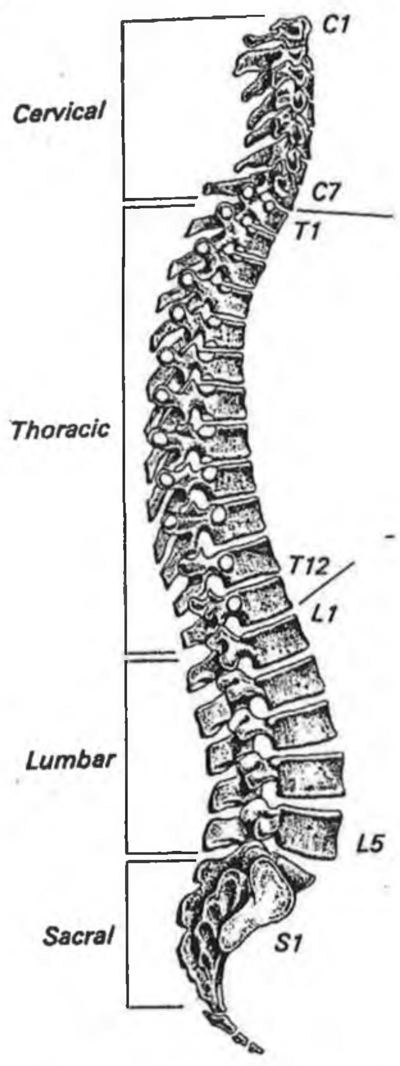


Figure 2-7: Vertebral column anatomy

The vertebral column of the articulated skeleton has four curves. They are thoracic, sacral, cervical and lumbar curves. The first two are concave forward and the later couple is convex forward. The first two are present at birth and do not alter in shape. The last two developed after birth.

The fibrocartilaginous intervertebral discs are very strong and can resist considerable vertical loading forces. Normally the vertebra fails and not the disc [150]. Carlsoo [151] estimated that the disc can withstand a load up to 6000N before damage occurs to the vertebral end plate. White and Panjabi [152] summarised the range of motion for each intervertebral junction of the spine in the sagittal, frontal and transverse planes from previous literatures. Table 2-1 shows the degree of rotation at vertebral junctions of the spine. Considerable variation from individual to another is expected.

Table 2-1: Degree of rotation at vertebral junctions of the spine [152]

| Vertebral junction | Sagittal plane (degree) | Frontal plane (degree) | Transverse plane (degree) |
|---|---|---|---|
| Occiput – C1 | 13 | 8 | 0 |
| C1 – C2 | 10 | 0 | 47 |
| C2 – C3 | 8 | 10 | 9 |
| C3 – C4 | 13 | 11 | 11 |
| C4 – C5 | 12 | 11 | 12 |
| C5 – C6 | 17 | 8 | 10 |
| C6 – C7 | 16 | 7 | 9 |
| C7 – T1 | 9 | 4 | 8 |
| T1 – T2 | 4 | 6 | 9 |
| T2 – T3 | 4 | 6 | 8 |
| T3 – T4 | 4 | 6 | 8 |
| T4 – T5 | 4 | 6 | 8 |
| T5 – T6 | 4 | 6 | 8 |
| T6 – T7 | 5 | 6 | 8 |
| T7 – T8 | 6 | 6 | 8 |
| T8 – T9 | 6 | 6 | 7 |
| T9 – T10 | 6 | 6 | 4 |
| T10 – T11 | 9 | 7 | 2 |
| T11 – T12 | 12 | 9 | 2 |
| T12 – L1 | 12 | 8 | 2 |
| L1 – L2 | 12 | 6 | 2 |
| L2 – L3 | 14 | 6 | 2 |
| L3 – L4 | 15 | 8 | 2 |
| L4 – L5 | 17 | 6 | 2 |
| L5 – S1 | 20 | 3 | 5 |

### 2.8.3 DATA FROM CADAVERS

Dempster [153] explained the planes of the dissections. The living body was submerged to the defined planes of the joint centres. His work was explicit and data were collected on the most cadavers. The density data were the only data used after obtaining water displacement volumes to calculate each segment weight of the living. Each segment's density thus corresponded to the volume of the segment. Krogman and Johnson [154] compared the cadaver data of Dempster to other three researchers, Braune and Fischer, Fischer, and Amar. Table 2-2 and Table 2-3 show the Dempster's planes and specific gravity of body segments. Later in 1965, Dempster and Gaughran [155] studied body constants of nine male white cadavers of normal appearance and average build. The limb data were supplemented by a further analysis of 11 upper and 41 lower limbs. Each cadaver was measured, weighed and somatotyped. Each segment was dissected into its component parts and these were weighed.

Table 2-2: Dempster's planes of joint center [148]

| | |
|---|---|
| Hand | • The mid-point of the pisiform bone<br>• The distal wrist crease at the palmaris longus tendon or the midpoint on the volar surface of the navicular bone<br>• The palpable sulcus dorsally between the lunate and capitate bones |
| Hand plus forearm | • The lower border of the medial epicondyle of the humerus<br>• Eight millimeters above the radiale |
| Whole upper limb | • The palpable sulcus above the acromioclavicular joint<br>• The anterior axillary fold at the projection of the thoracic contour<br>• The posterior axillary fold at the projection of the thoracic contour |
| Foot | • The superior border of the calcaneus anterior to the achilles tendon as palpated medially and laterally<br>• The upper border of the head of the talus<br>• The lower tip of the fibula |
| Foot and shank | • The mid point of the posterior curvature of the medial condyle of the femur as palpated<br>• The same for the lateral condyle |
| Whole lower limb | • The anterior superior spine of the test side<br>• The ischiopubic sulcus between the thigh and the scrotum<br>• The line from the ischium to above the femoral trochanter |
| Head and neck | • The top of the sterno-calvicular joint<br>• Between the 7th cervical and the 1st thoracic vertebra |
| Head | • Decapitate the skull from the atlas |
| Thorax | • The disc between the 10th and 11th thoracic bertebra<br>• The lowest fibers of the pectoralis major or the middle of the xiphoid process |
| Between pelvis and abdomen* | • The disc between the 3rd and 4th lumbar vertebra<br>• The umbilicus |

* Not dissected by Dempster [153]

Table 2-3: Specific gravity of body segments [153]

| | | | | | |
|---|---|---|---|---|---|
| Hand | 1.16 | Forearm | 1.13 | Upper arm | 1.07 |
| Foot | 1.10 | Shank | 1.09 | Thigh | 1.05 |
| Thorax | 0.92 | Abdomen | 1.01 | Pelvis | 1.01 |
| Head-Neck | 1.11 | | | | |

Contini [156], Drillis and Contini [157] and Santschi, Dubois and Omoto [158] have done work on the living, but all density and centre of gravity data were on the whole body. It is not possible to get accurate density data on living subjects, so cadaver data must be used. Clause *et al.* [159] quantified the discrepancy between the mid-volume level of a submerged limb and the actual centre of gravity. The centre of gravity moved distally from the mid-volume position with the proposed correction factors. Table 2-4 shows the correction factors of different segments.

Table 2-4: Water volume from the distal end to the centre of mass [159]

| | | | |
|---|---|---|---|
| Lower leg | 45.1% | Lower leg and foot | 46.4% |
| Thigh | 46.5% | Whole leg | 42.6% |
| Forearm | 43.7% | Forearm and hand | 45.2% |
| Upper arm | 45.6% | | |

## 2.8.4 BODY SEGMENT WEIGHTS

For body parts with a mass of less than 20 kg, critical weightings were made on a solution balance calibrated in grams and accurate with any combination of counterbalancing weights to within one or two grams by Dempster and Gaughran [155]. Larger masses such as those of the total trunk, or abdominopelvic mass, were measured to the nearest quarter-pound on a 300lb. Av. Fairbanks grain scale, then expressed in metric units. Total body mass was measured by placing the body on two bathroom scales. Data were collected and converted to metric units later. A measurement of ±1.0 percent may be assumed. Table 2-5 shows the mean weights of male cadaver segments by Dempster and Gaughran. Whereas Braune and Fischer's [160] classical data can be found in the following table. Clauser *et al.* [159] measured 13 cadavers body weight and Table 2-7 shows the findings.

Table 2-5: Segment weights by Dempster and Gaughran [155]

| Segment | Number of Samples | Mean weight (gm) | Number of Samples | % of total |
|---|---|---|---|---|
| Total body | 7 | 61190±8137 | | 100 |
| Head and trunk | 18 | 34637±5607 | 6 | 56.34±2.45 |
| Head and trunk minus shoulders | 18 | 28077±3994 | 5 | 46.02±2.239 |
| Head and neck | 16 | 5119±838 | 7 | 7.92±0.85 |
| Shoulders | 34 | 3401±843 | 14 | 5.27±0.546 |
| Thorax | 17 | 7669±2270 | 7 | 10.97±1.521 |
| Abdomino-pelvic headless trunk | 18 | 16318±2505 | 7 | 26.39±2.908 |
| Arm | 42 | 1636±350 | 14 | 2.64±0.294 |
| Forearm | 42 | 947±199 | 14 | 1.531±0.166 |
| Hand | 42 | 378.3±71.7 | 14 | 0.612±0.058 |
| Thigh | 41 | 609.6±985 | 14 | 10.008±1.197 |
| Shank | 41 | 2852±695 | 14 | 4.612±0.534 |
| Foot | 41 | 884±178 | 14 | 1.431±0.142 |

Table 2-6: Segment weights by Braune and Fischer [160]

| Segment | Number of Samples | Mean weight (gm) | Number of Samples | % of total |
|---|---|---|---|---|
| Total body | 5 | 58895 | | 100 |
| Head and trunk | 5 | 30824 | 5 | 52.21±2.97 |
| Arm | 22 | 2017±406 | 10 | 3.167±0.27 |
| Forearm | 18 | 1342±242 | 6 | 2.087±0.245 |
| Hand | 18 | 536.1±84.4 | 6 | 0.833±0.045 |
| Thigh | 22 | 6632±783 | 10 | 10.924±0.769 |
| Shank | 22 | 2924±379 | 10 | 4.680±0.353 |
| Foot | 22 | 1072±106 | 10 | 1.765±0.194 |

Table 2-7: Cadaver segment weights by Clauser *et al.* [159]

| Segment | % of total |
|---|---|
| Head | 7.3 |
| Trunk | 50.7 |
| Arm | 2.6 |
| Forearm | 1.6 |
| Hand | 0.7 |
| Total arm | 4.9 |
| Forearm and hand | 2.3 |
| Thigh | 10.3 |
| Shank | 4.3 |
| Foot | 1.5 |
| Total leg | 16.1 |
| Shank and foot | 5.8 |
| Total body | 100 |

Table 2-8: Segment weights as percentage of total body weight [148]

| | Men (Sample Size = 35) | | Women (Sample Size = 100) | |
|---|---|---|---|---|
| Segment | Mean (%) | SD* | Mean (%) | SD* |
| Hand | 0.65 | 0.06 | 0.5 | 0.026 |
| Forearm | 1.87 | 0.2 | 1.57 | 0.1 |
| Upper arm | 3.25 | 0.49 | 2.9 | 0.32 |
| Foot | 1.43 | 0.13 | 1.33 | 0.02 |
| Shank | 4.75 | 0.53 | 5.35 | 0.47 |
| Thigh | 10.5 | 1.21 | 11.75 | 1.86 |
| Whole trunk | 55.1 | 2.75 | 53.2 | 4.64 |
| Head and neck | 8.26 | | 8.2 | |
| Thorax | 20.1 | | 17.02 | |
| Abdomen | 13.06 | | 12.24 | |
| Pelvis | 13.66 | | 15.96 | |

Table 2-9: Trunk segment specific weights over trunk weight

| | Men (Sample Size = 35) | | Women (Sample Size = 100) | |
|---|---|---|---|---|
| Segment | Mean (%) | SD* | Mean (%) | SD* |
| Head and neck | 0.152 | 0.0196 | 0.148 | 0.0185 |
| Thorax | 0.358 | 0.011 | 0.319 | 0.0291 |
| Abdomen | 0.242 | 0.016 | 0.235 | 0.014 |
| Pelvis | 0.255 | 0.018 | 0.309 | 0.015 |
| Abdomen and pelvis | 0.496 | 0.021 | 0.4883 | 0.0245 |

* Standard Deviation

Plagenhoef *et al.* [148] performed the water immersion technique on 135 college-age athletes, which consisted of 100 female and 35 male. The sample produced total weight of each extremity and the trunk as one segment. Table 2-8 shows the segment weights as percentage of total body weight.

From the same sample, Plagenhoef *et al.* selected 18 women and 15 men at random to perform segment analysis of the trunk. The cadaver dissection of the trunk by J.L. Parks [148] was used as a guideline for the water displacement landmarks. Table 2-9 shows the trunk segments specific weight ratio available from [148].

Table 2-10: Endpoint for segment length measurements in Dempster and Gaughran study [155]

| Body part | Superior (proximal) end point | Inferior (distal) end point |
|---|---|---|
| Shoulder | Centre of articular face of sternal end of clavicle | Centre of curvature of humeral head (proximal surface of cut that bisects midrange joint angle) |
| Arm | Same as shoulder distal end point (distal face of cut) | Junction, transverse axis of trochlea of humerus at narrowest cross section of ulnar articulation |
| Forearm | Same as arm distal end point (distal face of cut) | Centre of curvature of proximal end of capitate bone (proximal face of cut) |
| Hand | Same as forearm distal end point (distal face of cut) | Proximal interphalangeal knuckle of finger II, position of rest |
| Thigh | Centre of curvature of femoral head | Middle of a line through the centre of curvature of the posterior aspect of the two femoral condyles (proximal face of cut) |
| Shank | Same as thigh distal end point (distal face of cut) | Centre of the area of the cut body of the talus (proximal face of cut) |
| Foot | Same as shank distal end point (distal face of cut) | Tip of great toe |
| Entire head and trunk | Vertex of head | Midpoint of the transacetabular line |
| Head and neck | Vertex of head | Centre of cut intervertebral disc at lower face of centrum of vertebra C-7 |
| Thorax | Midpoint of disc tissue at superior face of centrum of vertebra T-1 | Midpoint of disc tissue of inferior face of centrum of vertebra T-12 |
| Abdomino-pelvic segment | Midpoint of disc tissue, superior face of centrum of vertebra L-1 | Same as entire head and trunk distal end point |
| Transhumeral width | Span between shoulder distal end point of right and left sides | |
| Transacetabular width | Span between centres of femoral heads of right and left sides, measured after removal of lower limbs | |
| Hip-to-shoulder height | Vertical distance between transhumeral and transacetabilar vertical height. | |

## 2.8.5 BODY LENGTHS

Dempster and Gaughran measured the body parts in metric system using a standard anthropometer[*] for larger measurements [155]. Table 2-10 shows the end-points used for calliper measurements relating to the limb segments after they were separated into units. Measurement of the segment lengths can be found in Table 2-11. Whereas Braune and Fischer's classical data [160] can be found in the following table. An average set of segment lengths expressed as a percent body height has been prepared by Drillis and Contini [161] and is shown in Figure 2-8.

Table 2-11: Segment lengths by Dempster and Gaughran [155]

| Segment | Sample Size | Length (cm) | % in relation to stature |
|---|---|---|---|
| Thigh | 18 | 40.56±2.42 | 23.99±1.43 |
| Shank | 18 | 42.36±2.07 | 25.05±1.22 |
| Foot | 18 | 6.23±1.22 | 3.694±0.720 |
| Lumbocoxal | 8 | 28.99±3.40 | 17.14±2.01 |
| Thoracic | 8 | 26.57±2.54 | 15.71±1.50 |
| Craniocervical | 8 | 13.71±2.21 | 8.11±1.31 |
| Arm | 18 | 29.34±1.67 | 17.35±0.99 |
| Forearm | 18 | 26.58±0.87 | 15.72±0.52 |
| Hand | 18 | 5.777±0.745 | 3.422±0.441 |
| Transpelvic | 19 | 16.32±1.79 | 9.82±1.17 |

Table 2-12: Segment lengths by Braune and Fischer [160]

| Segment | Sample Size | Length (cm) | % in relation to stature |
|---|---|---|---|
| Thigh | 10 | 43.32±1.21 | 24.26±0.74 |
| Shank | 10 | 40.55±1.43 | 24.72±0.87 |
| Craniocervical | 1 | - | 10.6 |
| Arm | 6 | 29.41±1.12 | 17.93±0.68 |
| Forearm | 4 | 27.42±1.25 | 16.72±0.76 |
| Hand | 6 | 5.38±0.33 | 3.28±0.20 |
| Transpelvic | 3 | 16.73±0.48 | 10.20±0.29 |

[*] A spreading calliper for measurements of parts of intermediate size and a sliding calliper for measurements of small parts (hand and foot).

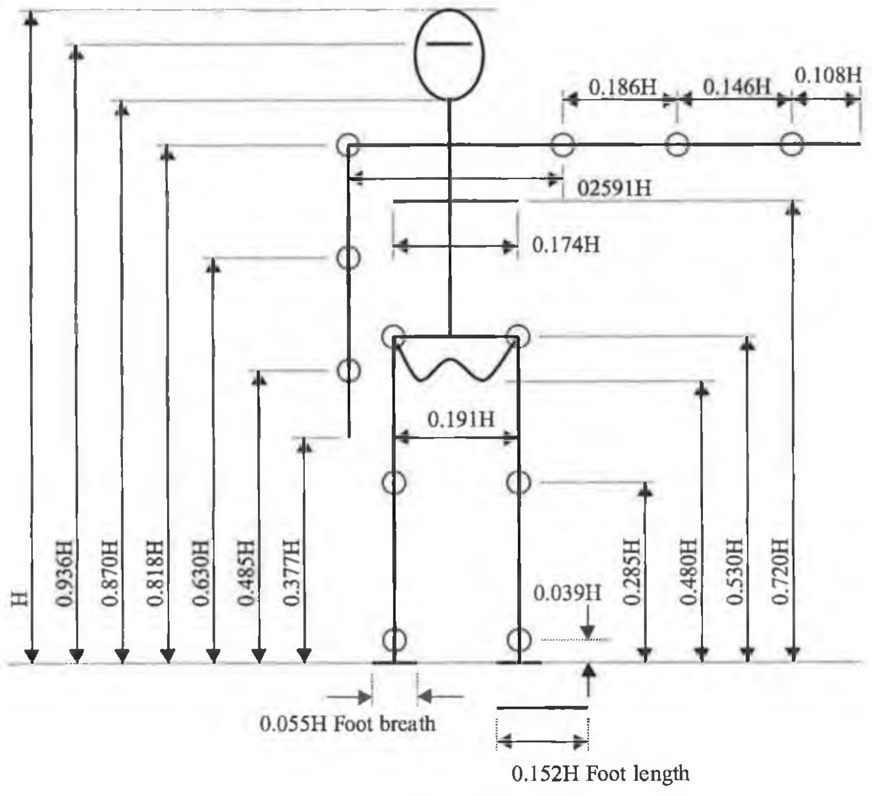


Figure 2-8: body segment lengths expressed as a fraction of body height, H [161]

Plagenhoef *et al.* [148] measured segment lengths of 35 men and 73 women. The length is tabulated in Table 2-13 as a percentage of the height so these data may be used on any size person. The average standard of deviation was one over every 16 centimetres measurement, which approximately 6%. This is a very small variation in both segment lengths and segment weights when expressed as a percentage of height and weight.

## 2.9 HUMAN NUMERICAL MODELS

Human body dynamic response under severe acceleration/deceleration has been the topic of extensive investigation. It is due to the resulting injuries and fatalities, as well as complexity of the human body. Various computer-based mathematical models have been developed and attained wider acceptance nowadays. These models are used to predict human dynamic motion response during crash as an attractive supplement to full-scale experimental testing. According to reference [162], McHenry is one of the first to proposed a mathematical simulation model to describe the dynamic response of a vehicle occupant involved in collision event in 1963. Since then, many more sophisticated models have been developed for use as simulators of occupant kinematics. Nowadays, it becomes a crucial part of the field of automotive safety.

Table 2-13: Segment length as a percentage of total height [148]

| Segment | Men (Sample Size=35) (%) | Women (Sample Size=73) (%) |
|---|---|---|
| Hand* | 5.75 | 5375 |
| Forearm | 15.7 | 16.0 |
| Upper arm | 17.2 | 17.3 |
| Foot* | 4.25 | 4.25 |
| Shank | 24.7 | 25.0 |
| Thigh | 23.2 | 24.9 |
| Trunk* | 30.0 | 29.0 |
| Head and neck* | 10.75 | 10.75 |
| Thorax | 12.7 | 12.7 |
| Abdomen | 8.1 | 8.1 |
| Pelvis | 9.3 | 9.3 |
| Shoulder to shoulder† | 24.5 | 20.0 |
| Hip to hip | 11.3 | 12.0 |

King and Chou [163] reviewed several gross motion simulators, which including CAL3D and MVMA2D, in 1976. Prasad explicated the formulation of CAL3D, MVMA2D and MADYMO2D‡ models in reference [164]. Prasad and Chou [162] reviewed some simple one-dimensional models for side impact simulations, more complicated 2D and 3D models. Recent development of CAL3D [165], MADYMO2D/3D [166], UCIN3D [167] and SOMLA [168] were discussed. The total body is typically described as a set of rigid segments linked by kinematic joints in the computer simulation. The human dynamic simulation code of CAL3D is based on Articulated Total Body (ATB) model. The equations of motion are neither Newtonian

* to center of gravity

† gleno-hum

‡ developed by TNO in Holland in 1979.

nor Lagrangian. It is similar to the Newtonian method and the constraint forces are explicitly contained in the equations of motion without employing Lagrange multipliers. Multiple tree structures composed of rigid bodies connected by joints are used in MADYMO code. The equations of motion are derived using Lagrangian methods. The dynamic analysis of the UCIN3D model is based on a virtual work type principle called Lagrange's form of d'Alembert principle, which is claimed by the developers to have the advantages of both the Lagrangian and Newtonian but without the associated disadvantage [162]. SOMLA combines a three-dimensional occupant model and a finite element seat model to interact dynamically. Lagrangian formulation is used in the analytical treatment for deriving the equations of motion of sets of rigid bodies.

## 2.10 RESISTIVE MOMENT OF THE HUMAN JOINTS

Detailed description of joint kinematics and resistive torque for human crash simulations is very limited [169]. CAL3D use the Generator of Body Data (GEBOD) program [170], while MADYMO utilises an adapted version of GEBOD for geometric and inertial properties and dummy joint model parameters for human crash simulations. SOMLA employs Glanville and Kreezer's joint data to model the limits of joint motion for both voluntary and forced rotations [171, 172].

Engin [173] conducted human joint properties tests at Ohio State University. The motion of a moving body segment with respect to the fixed one was monitored. The force and moment vectors causing the motion were recorded simultaneously. Three male volunteers were involved. The study was on shoulder, hip, knee, elbow and ankle joints, where voluntary and forced range of motion, passive resistive forces and moments and the twist resistive torques were measured. The main components of the testing apparatus were a subject restraint system, a global force applicator and an exoskeletal device. Later, Engin and Chen [174] utilised a sonic digitising technique to measure the joints. Ten male volunteers were involved for testing of their voluntary and forced range of motion, and passive resistive torque on shoulder, hip and elbow joints.

## 2.10.1 MA *ET AL.*'S RESISTIVE TORQUES FORMULATION

The above mentioned data were used by Ma *et al.* [169] to formulate resistive torques for various joints. Figure 2-9 shows the local and joint co-ordinate systems of Ma *et al.*'s model.

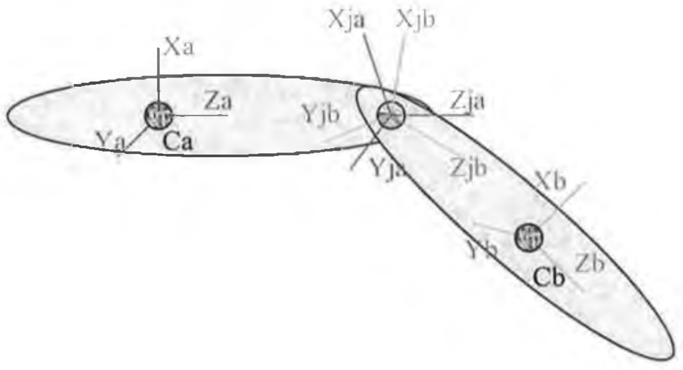


Figure 2-9: Local and joint co-ordinate systems [169]

Local co-ordinate systems (Xa, Ya, Za and Xb, Yb, Zb) had origins at the segment centres of gravity (Ca and Cb). The orientation of a local co-ordinate system was defined so that when the human body is in an upright standing posture, with the arms at the side, the local X axes were directed back to front, the Y axes were directed left to right and the Z axes were directed downward. Two joint co-ordinate systems were defined for each joint. Joint co-ordinate Xja, Yja, Zja was rigidly attached to a proximal body segment a, while the other set to the distal body segment b. The origin of each joint co-ordinate system was specified in the local co-ordinate systems and calculated using regression equations based on anthropometric survey and stereophotometric studies or reference [175]. The orientations of the joint co-ordinate systems were specified using rotation angles, yaw, pitch and roll, with respect to the local co-ordinate systems. Table 2-14 shows the orientations of the joint co-ordinate systems for different joints.

A pin joint between two segments has only one relative degree of freedom. Pin joint was characterised with the joint location, orientation of the joint, co-ordinate systems with respect to local co-ordinate systems, and maximum voluntary flexion/extension (or stop) angles $\theta_f$ and $\theta_e$ as shown in Figure 2-10. A polynomial relationship was shown to describe moment characteristics of a pin joint beyond the maximum voluntary range of motion and yield the resistive flexion and extension resistive torques ($T_f$ and $T_e$) in the Equation (2-2). Table 2-15 shows the respective stop angles and coefficients of resistive torque functions for elbow and knee.

$$T(\theta) = \alpha_1(\theta - \theta_s) + \alpha_2(\theta - \theta_s)^2 + \alpha_3(\theta - \theta_s)^3 + \ldots \alpha_n(\theta - \theta_s)^n \qquad (2\text{-}2)$$

where

| | |
|---|---|
| $\theta$ | Flexion or extension angle of the joint |
| $\theta_s$ | Flexion or Extension stop angle |
| $\alpha_I$ | Coefficients determined by regression analysis of the human joint tests; $i$ = 1..n |

Table 2-14:Orientations of the joint co-ordinate systems in degree [169]

| | Proximal joint axes orientation | | | Distal joint axes orientation | | |
|---|---|---|---|---|---|---|
| Joint | Yaw | Pitch | Roll | Yaw | Pitch | Roll |
| Right elbow | -15 | 65 | 0 | 15 | 0 | 0 |
| Left elbow | 15 | 65 | 0 | 15 | 0 | 0 |
| Right knee | 0 | -66 | 0 | 0 | 0 | 0 |
| Left knee | 0 | -66 | 0 | 0 | 0 | 0 |
| Right shoulder | 59.3 | 79.1 | 0 | 0 | 0 | 0 |
| Left shoulder | -59.3 | 79.1 | 0 | 0 | 0 | 0 |
| Right hip | 14 | 48 | 0 | 0 | 0 | -7 |
| Left hip | -14 | 48 | 0 | 0 | 0 | 7 |
| Right ankle | 0 | 65 | 0 | 0 | 0 | 0 |
| Left ankle | 00 | 65 | 0 | 0 | 0 | 0 |

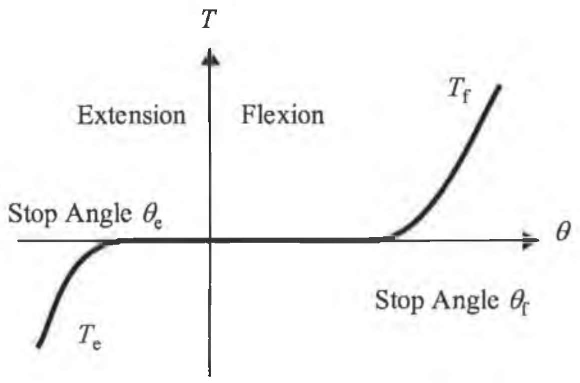


Figure 2-10: Pin joint resistive torque model [169]

Table 2-15: Stop angles and coefficient of resistive torque functions for pin joints [169]

| | | $\alpha_n$ (N-m/rad$^n$) | | |
|---|---|---|---|---|
| Joint | $\theta_s$ | n=1 | n=2 | n=3 |
| Elbow (extension) | 30° | 10.20 | 30.68 | 24.72 |
| Elbow (flexion) | 34° | 6.82 | 14.88 | 2.17 |
| Knee (extension) | 23° | 5.90 | 49.83 | 19.92 |
| Knee (flexion) | 34° | 17.39 | 46.14 | 28.31 |

A ball-and-socket joint has three angular degrees of freedom. They are flexure, azimuth and twist, which are based on the relative orientation of the two joint co-ordinate systems. It is shown in Figure 2-11. The flexure angle θ is the angle between the two Z-axes, and the azimuth angle ϕ is the angle between the base $X_{ja}$ axis and the projection of the $Z_{jb}$ axis into the $X_{ja}$-$Y_{ja}$ plane. The range for θ and ϕ is 0° to 180° and 0° to 360° respectively. $X_{ja}$, $Y_{ja}$, $Z_{ja}$ define the co-ordinate system of segment "a" connected to segment "b" with joint "j".

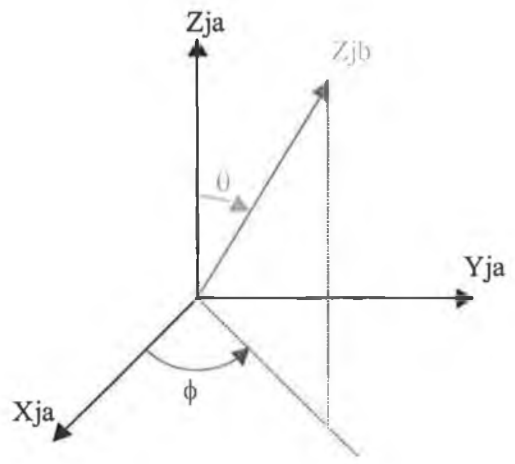


Figure 2-11: Flexure And Azimuth Angles For A Ball-And-Socket Join

The maximum voluntary range of motion was defined as a contour dependent on the azimuth angle described by the following equation.

$$\theta_s(\phi) = \sum_{i=1}^{5} \cos^{n-1}\phi(\xi_{2n-1} + \xi_{2n}\sin\phi) \tag{2-3}$$

Where

| | |
|---|---|
| $\theta_s$ | Flexure stop angle |
| $\phi$ | Azimuth angle |
| $\xi_i$ | Coefficients based on the regression analysis of the joint tests $i = 1..10$ |

The resistive joint torque was measured as the joint was pushed beyond the stop angle $\theta_s$ at a constant azimuth angle $\phi$, until the subject experienced discomfort. The tested joint resistive torques were fitted into the following equation.

$$\begin{aligned} T(\theta,\phi) = {} & (\eta_1 + \eta_2\cos\phi + \eta_3\sin\phi)\theta + \\ & (\eta_4\cos^2\phi + \eta_5\cos\phi\sin\phi + \eta_6\sin^2\phi)\theta^2 \ldots \\ & \ldots + (\eta_7\cos^3\phi + \eta_8\cos^3\phi\sin\phi + \eta_9\cos\phi\sin^2\phi + \eta_{10}\sin^3\phi)\theta^3 \\ T(\theta,\phi) = {} & 0, (if\, \theta \le \theta_s) \end{aligned} \tag{2-4}$$

Where

| | |
|---|---|
| $T$ | Function of the passive resistive torques acting on the joints |
| $\eta_I$ | Coefficients based on the regression analysis of the joint tests $i = 1..10$ |

The following tables are the stopping angles and coefficients of flexure polynomials for right shoulder, right hip and ankle joint respectively. Viscous Damping was used. 1.942 Nm-sec/rad viscous damping for elbow, knee, hip and ankle joints. 3.884 Nm-sec/rad viscous damping for shoulder joint.

Table 2-16: Stop angles and coefficient of flexure polynomials for right shoulder [169]

| | | $\alpha_n$ (N-m/rad$^n$) | | |
|---|---|---|---|---|
| $\phi$ (°) | $\theta_s$ (°) | n=1 | n=2 | n=3 |
| 0 | 84.04 | 25.42 | 13.71 | -1.36 |
| 30 | 80.09 | 22.13 | 10.84 | -2.40 |
| 60 | 77.25 | 23.09 | 13.55 | -1.32 |
| 90 | 80.60 | 21.38 | 13.26 | -0.58 |
| 120 | 99.29 | 21.27 | 8.51 | -1.18 |
| 150 | 84.87 | 25.43 | 13.26 | -0.73 |
| 180 | 60.07 | 26.40 | 23.99 | 1.36 |
| 210 | 65.91 | 36.87 | 29.19 | 2.40 |
| 240 | 82.41 | 40.25 | 24.60 | 1.32 |
| 270 | 91.21 | 30.74 | 18.47 | 0.57 |
| 300 | 89.27 | 29.21 | 19.90 | 1.15 |
| 330 | 89.62 | 32.62 | 19.93 | 0.73 |

Table 2-17: Stop angles and coefficient of flexure polynomials for right hip [169]

| | | $\alpha_n$ (N-m/rad$^n$) | | |
|---|---|---|---|---|
| $\phi$ (°) | $\theta_s$ (°) | n=1 | n=2 | n=3 |
| 0 | 63.85 | 93.35 | 76.40 | -0.032 |
| 30 | 58.28 | 93.35 | 76.40 | -0.032 |
| 60 | 38.91 | 93.35 | 76.40 | -0.032 |
| 90 | 36.17 | 93.35 | 76.40 | -0.032 |
| 120 | 39.74 | 93.35 | 76.40 | -0.032 |
| 150 | 50.86 | 93.35 | 76.40 | -0.032 |
| 180 | 63.37 | 93.35 | 76.40 | -0.032 |
| 210 | 47.67 | 93.35 | 76.40 | -0.032 |
| 240 | 37.64 | 93.35 | 76.40 | -0.032 |
| 270 | 34.86 | 93.35 | 76.40 | -0.032 |
| 300 | 38.25 | 93.35 | 76.40 | -0.032 |
| 330 | 54.14 | 93.35 | 76.40 | -0.032 |

Table 2-18: Stop angles and coefficient of flexure polynomials for ankle joint [169]

| | | $\alpha_n$ (N-m/rad$^n$) | | |
|---|---|---|---|---|
| $\phi$ (°) | $\theta_s$ (°) | n=1 | n=2 | N=3 |
| 0 | 4.0 | 18.95 | -1.37 | 15.03 |
| 180 | 4.0 | 20.23 | 7.76 | 15.12 |

## 2.11 MOTORCYCLE CRASH

Motorcycling is a rapidly increasing mode of transport in developing countries, and continuous to be a very import mode in many industrialised nations. It is well known that the mode carries a very high accident risk and yet almost the only protection afforded to a motorcyclist is the helmet. Table 2-19 shows motorcyclist fatalities of several nations in 1980 and 1990 [176]. Table 2-20 shows a 21 percent increase in the European countries for which data is available.

As a whole, the number of motorcycle fatalities decreased. How fatalities among other types of road users are declining more quickly, so those motorcycle fatalities from an increasing proportion of all road deaths in most countries. The increasing proportion of casualties should be seen against an increasing trend in motorcycle ownership in most countries.

Table 2-19: Motorcyclist fatalities in various countries

| | Number of fatalities | | | As a %age of all road fatalities | | |
|---|---|---|---|---|---|---|
| Country | 1980 | 1990 | % change | 1980 | 1990 | % change |
| Australia | 442 | 263 | -41 | 13.5 | 10.5 | -16 |
| Austria | 106 | 107 | +1 | 7.3 | 7.2 | -1 |
| Belgium | 170 | 106 | -38 | 7.1 | 5.4 | -24 |
| Denmark | 59 | 39 | -34 | 8.6 | 6.2 | -28 |
| Finland | 21 | 28 | +33 | 3.8 | 4.3 | +13 |
| France | 1136 | 1031 | -9 | 8.5 | 9.2 | +8 |
| Germany | 1232 | 769 | -38 | 10.5 | 9.6 | -8 |
| UK | 1113 | 621 | -44 | 17.8 | 11.7 | -34 |
| Greece | 106 | 183 | +73 | 7.7 | 13.4 | +73 |
| Ireland | 48 | 41 | -15 | 8.5 | 8.6 | +1 |
| Italy | 822 | 706 | -45 | 9.0 | 10.0 | +11 |
| Japan | 1163 | 1920 | +65 | 9.6 | 13.2 | +37 |
| Netherlands | 130 | 72 | -45 | 6.5 | 5.2 | -20 |
| New Zealand | 91 | 114 | +25 | 15.2 | 15.6 | +3 |
| Norway | 29 | 25 | -45 | 8.0 | 7.5 | -6 |
| Spain | 316 | 792 | +151 | 4.8 | 8.8 | +81 |
| Sweden | 43 | 46 | +7 | 5.1 | 6.0 | +18 |
| Switzerland | 139 | 160 | +15 | 11.2 | 16.8 | +50 |
| USA | 5079 | 3173 | -38 | 9.8 | 7.0 | -29 |
| Total / Average | 12245 | 10196 | -17, | 9.1 | 9.3 | +2 |

Table 2-20 shows a drop of death percentage from 1980 to 1990. Germany shows a large fall in the fatality rate and was ascribed partly to an increase in driver age consequent on changes in licensing requirement [176]. Improved education and training, enforcement of helmet wearing, the use of headlights in the daytime, better braking, tyres and machine stability also contributed to the drop of the overall fatality rate. The above table also shows the motorcycle fatality rate was 2.8 times higher than car, means 2.8 times more dangerous to be used compare to car. The figures underestimated the higher risk of motorcycles, since it takes no account of the smaller average distance travelled by motorcycles than by cars in a specific period. The fatality rate per vehicle-km of motorcycle to car was obtained to have a clearer picture. The rate is several times higher comparing to car. The uncompleted statistic data of fatality rate per vehicle-km of motorcycle to car shows an average of 10 times higher risk of riding a motorcycle comparing to car.

Table 2-20: Motorcyclist ownership change, fatality rates and comparison with car fatality rates in various countries [176]

| | Change in motorcycle ownership | Death per 100000 registered motorcycle | | | Ratio of death rates motorcycle to car in 1990 | |
|---|---|---|---|---|---|---|
| Country | 1980-90 (%) | 1980 | 1990 | % change | Per veh* | Per veh-km |
| Australia | -2 | 100 | 60 | -39 | 4.2 | 10.1 |
| Austria | +15 | 117 | 102 | -13 | 3.0 | 7.3 |
| Belgium | +22 | 150 | 77 | -49 | 2.5 | |
| Denmark | +28 | 172 | 88 | -49 | 5.0 | 10.8 |
| Finland | +38 | 48 | 47 | -3 | 2.8 | |
| France | +22 | 182 | 136 | -25 | 5.0 | |
| Germany | +92 | 167 | 54 | -67 | 2.2 | 11.0 |
| UK | -31 | 108 | 87 | -19 | 8.6 | 14.7 |
| Greece | +40 | 88 | 108 | +23 | 2.7 | |
| Ireland | -9 | 169 | 157 | -7 | 6.9 | 15.9 |
| Italy | +37 | 75 | 47 | -37 | 2.3 | |
| Japan | +34 | 37 | 45 | +24 | 2.5 | |
| Netherlands | +51 | 121 | 44 | -63 | 3.5 | 7.3 |
| New Zealand | -14 | 73 | 133 | +81 | 4.8 | |
| Norway | +5 | 100 | 82 | -17 | 6.3 | |
| Spain | -13 | 26 | 74 | +187 | 1.9 | |
| Sweden | -85 | 79 | 46 | -42 | 3.5 | |
| Switzerland | +120 | 101 | 53 | -48 | 3.7 | 5.2 |
| USA | -28 | 90 | 78 | -14 | 4.4 | |
| Total / Average | +9 | 110 | 89 | -20 | 2.8 | 10.3 |

* Vechile

Motorcycle is generally light in relation to the object it is impacting, and loses stability once struck, the dynamics of motorcycle collisions tend to be more complex than for four wheel vehicles. The rider's trajectory is more complex and variable than that of a vehicle occupant. After impact the bike tends to rotate about the impact point [182].

### 2.11.1 OBJECTS STRUCK

Several researchers conducted studies on the object struck by motorcycle in 1981. They are Harms[177], Hurt *et al.* [178], Kalbe *et al.* [179], and Otte *et al.* [180]. The summary of their findings is shown in the following table.

Table 2-21: Frequency of objects struck in motorcycle accidents

| | | Collision with (in %) | | | | |
|---|---|---|---|---|---|---|
| Study | Sample size | Car | Other 4 wheel | Other motorcycle | Fell off | Other obstacle |
| Harms | 766 | 41 | 6 | 2 | 30 | 6 |
| Hurt *et al.* | 900 | 65 | 5 | 3 | 19 | 6 |
| Kalbe *et al.* | 123 | 57 | 6 | 2 | 18 | 1 |
| Otte *et al.* | 272 | 68 | | - | 18 | 14 |

Vallee *et al.* [181] showed that although the highest proportion of collisions were with cars, only 33 percent of the objects struck by the rider head. This is because the rider's head often does not strike the collision vehicle, but the trajectory of the rider after collision brings the head into contact with other objects, often the road, motorcycle or roadside furniture [182].

### 2.11.2 COLLISION SPEED

The risk and seriousness of all types of injury has a direct proportional relationship with impact speed. A number of studies showed that the mean motorcycle speed is not very high, and in the range from 30 to 45 km/h [178, 180, 183, 184]. Figure 2-12 shows the distribution of motorcycle impact speeds by Otte in reference [185] with a sample size of 263.

Otte *et al.* reported very similar median speeds for both motorcycle and other vehicles in reference [180]. Both references of [183, 184] reported that the speeds of the other vehicles were generally substantially less than the motorcycle speed. Injury to middle facial skill occurs only in the higher speed ranges of over 40 km/hour. For impacts where contact was oblique from below the head and the lower jaw was fractured without injury to the body middle facial skull, the minimum relative speed was estimated to be 35km/h, whereas for a lower jaw fracture with middle-face fractures a minimum speed of 45km/h is necessary. However, these conclusions about the effect of speed on injury are based on small samples, and those results should be viewed with caution.

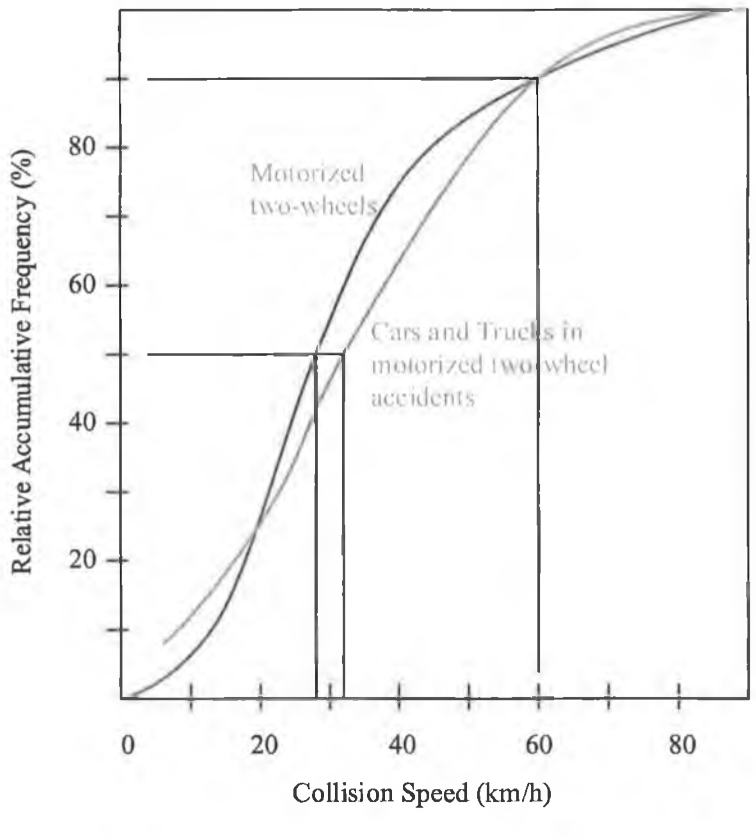


Figure 2-12: Distribution of motorcycle impact speeds

### 2.11.3 COLLISION MODES AND DIRECTIONS

The collision dynamic responses for both the vehicles and human body are dependent on the forces exerted. The forces are triggered either by the motorcyclist's kinetic energy or by the accident opponent's kinetic energy or both. In addition, the nature of impact (the hitting object and target obstacle collision mode – stiff or deformable impact) contributes to severity of a collision. From this point of view, two accident mechanisms can be categorised. They are type I – frontal collision, and type II – side collision. Type I collision has been defined in reference [182] as the group in which the kinetic energy of the motorcyclist in the direction of impact is greater than that of the accident opponent. These are often where the motorcycle impacts into the side of the other vehicle. The component of speed of the motorcycle along the line of impact is always greater than that of the accident opponent, regardless of the opponent's speed along its own line of motion. Type II collisions result from the converse of the definition of type I. A great number of references [183, 186, 187, 188] show that type I collisions contain a majority of accidents in which a frontal collision of the motorcycle took place. When colliding with four-wheelers, Otte *et al.* [189] categorised the interaction and showed that only 8.8% of the collisions where the rider flew over without impact. Another 11.4% of the riders landed upon the vehicle without full impact. Direct impact suffered highest frequency.

The great majority of collisions involve an impact to the front of the motorcycle. Both Otte *et al.* [190] and Whitaker [183] concluded that over 80 percent of impact are within ±20 degrees of the front. Later, Harms [191] showed 72 percent within ±15 degrees. The accident investigators tend to categorize impact directions by clock face angles, which limits the accuracy of angular information to 30-degree slots [182]. Sporner *et al.* [192] found high density of impact collisions around the front of motorcycle. The angular distribution of Sporner *et al.*'s findings are shown in Figure 2-13.

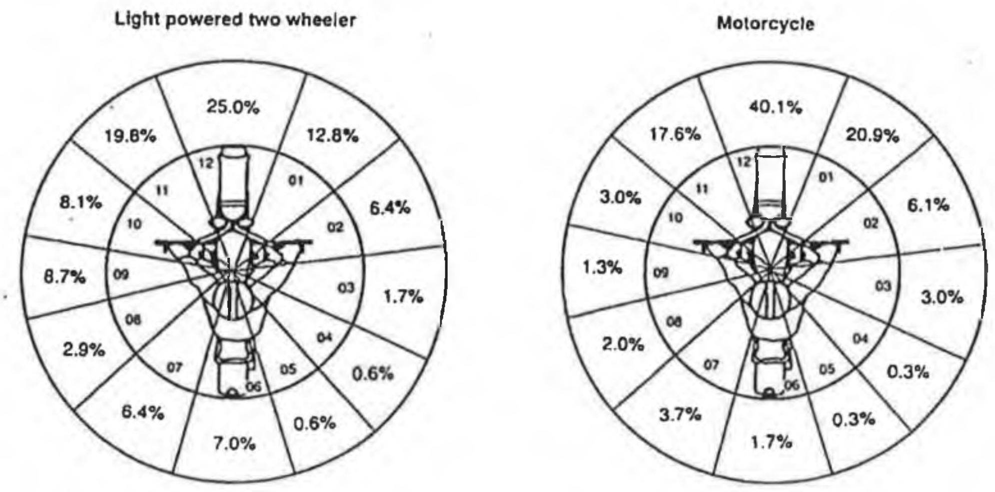


Figure 2-13: Angular distribution of collisions

## 2.11.4 MOTORCYCLE NUMERICAL MODELLING

Only a small amount of literatures in this area are available through the search conducted by the author. Thomson and Rathgeber [193] presented a vehicle model generator, which include motorcycle model. Two of the Newton-Euler formalism based methods were used to help generate non-linear or linearised equations of motion of the model. Happian-Smith *et al.* [194] showed a mathematical model of the motorcycle and rider. The model is shown in Figure 2-14, which was a typical multibody system consisting rigid bodies connected with joint and spring-damper elements. The paper shows the only items on motorcycle that absorb energy during head-on impact are the front wheel and forks. The authors suggested that the optimisation of these energy absorption characteristics would greatly aid the optimisation of other safety items, such as air-bag, to retard the rider at a safer level during impact.

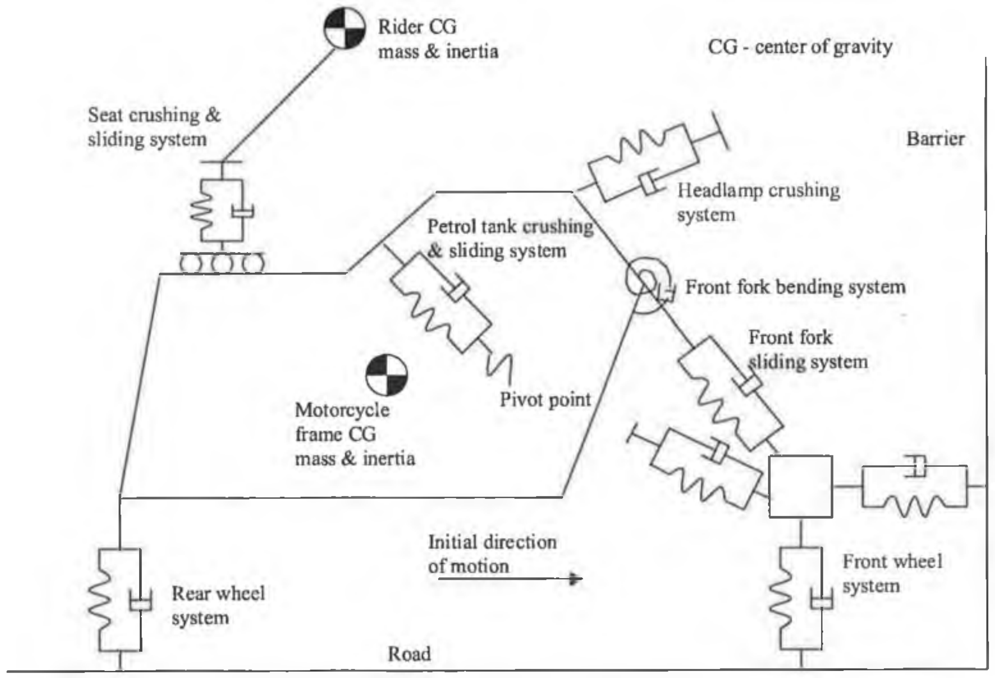


Figure 2-14: Happian-Smith *et al.*'s Motorcycle – rider model [194]

Chinn *et al.* [195] suggested a simple two-dimensional rigid body model in 1989. The model was made of a motorcycle running into a flat rigid barrier inclined at an angle to its original direction of travel. Translational and rotational displacements of the centre of gravity were derived and compared with results from full-scale crash tests. Different sizes of motorcycle were considered, with or without leg protecting fairings. Lately, Yettram *et al.* [196] proposed an articulated rigid multibody system to model the motorcycle with rider impact problem. The kinematic model of the motorcycle consisted of four rigid components connected by three compliant joints. The frame had six degrees of freedom, being free to translate and rotate relative to all three global Cartesian axes.

## 2.12 HUMAN INJURIES IN MOTORCYCLE CRASH

Different injury criterions were being suggested to predict mean potential injury. The criterions are used as a safety consideration in design of devices. Safety "level" can be measured and comparison can then be made. Injury distribution of 153 motorcyclists were organised by Otte *et al.* and reported in reference [189]. Reference [197], which summarised other investigators' data, shows Otte *et al.*'s finding is fairly characteristic of the other findings. Figure 2-15 shows the Otte *et al.*'s categorisation of human body in injury distribution studies. Table 2-22 shows the corresponding injury distribution.

The figure shows that the most frequently injured parts of the body are the legs (39 percent), the head (23 percent) and the arms (19 percent). 80 percent of casualties suffered some injury to legs, 56 percent to arms and 48 percent to head. However, the head injuries are more serious compared to legs and arms. Average AIS (Abbreviated Injury Scale, refer to the following section) scores of 2.4, 1.9 and 1.5 for head, leg and arm injuries respectively [182].

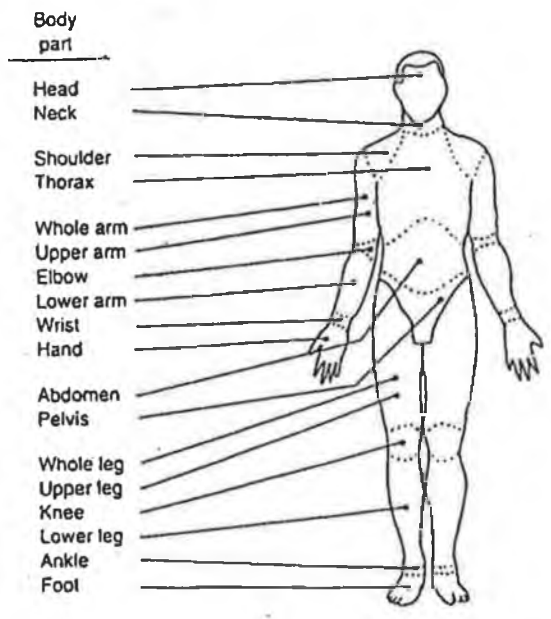


Figure 2-15: Otte *et al.*[189]'s categorisation of human body in injury distribution studies

Table 2-22: Distribution of injuries by body part

| Body part | % of all injuries | All injuries | Soft tissue | Fracture | Organs |
|---|---|---|---|---|---|
| Head | 22.9 | 48.4 | 39.6 | 14.4 | 29.8 |
| Neck | 1.8 | 8.7 | 5.7 | 3.3 | 0.4 |
| Shoulder | 3.8 | 19.0 | 12.5 | 9.0 | |
| Thorax | 6.7 | 19.0 | 12.3 | 8.5 | 6.3 |
| Whole arm | 18.8 | 56.2 | 50.6 | 17.1 | |
| Upper arm | 2.5 | 12.9 | 10.1 | 4.2 | |
| Elbow | 2.5 | 14.5 | 13.4 | 1.7 | |
| Lower arm | 4.6 | 21.9 | 14.5 | 8.8 | |
| Wrist | 2.0 | 11.8 | 9.9 | 1.7 | |
| Hand | 6.8 | 30.2 | 27.3 | 5.2 | |
| Abdomen | 4.0 | 13.6 | 8.3 | 1.1 | 6.6 |
| Pelvis | 3.1 | 13.1 | 11.0 | 3.3 | 0.3 |
| Whole leg | 38.9 | 81.2 | 75.7 | 32.2 | |
| Upper leg | 7.6 | 33.5 | 25.6 | 12.9 | |
| Knee | 11.3 | 49.7 | 48.1 | 3.1 | |
| Lower leg | 11.8 | 45.7 | 35.7 | 18.6 | |
| Ankle | 4.6 | 23.0 | 20.3 | 4.2 | |
| foot | 3.3 | 16.2 | 14.9 | 3.5 | |

### 2.12.1 ABBREVIATED INJURY SCALE

The Abbreviated Injury Scale (AIS) was introduced in January 1968. It has been widely used by the Medical Engineering Accident Investigation Teams of the NHTSA[*], NATO[†] Country Teams in Europe, General Motors, and AMA[‡] Physician-Police Teams [198]. AIS is a widely used scale in the world nowadays as an index to classify traumatic injuries. The AIS ratings are based on multiple criteria, which commonly used are energy dissipation (ED) and threat to life (TL). Other criteria, occasionally but not uniformly used are permanent impairment (PI), treatment period (TP) and incidence (IN).

[*] National Highway Traffic Safety Administration, USA

[†] North Atlantic Treaty Organization

[‡] American Medical Association

Table 2-23: Injury scaling in quantitation terminology

| Criteria | Code | Severity |
|---|---|---|
| ED (energy dissipation) | 1 | Little |
| | 2 | Minor |
| | 3 | Moderate |
| | 4 | Major |
| | 5 | Maximum |
| TL (threat to life) | 1 | None |
| | 2 | Minor |
| | 3 | Moderate |
| | 4 | Severe (serious) |
| | 5 | Maximum |
| PI (permanent impairment) | 1 | 20% or less |
| | 2 | 21-40% |
| | 3 | 41-60% |
| | 4 | 61-90% |
| | 5 | 90-100% |
| TP (treatment period) | 1 | 2 weeks or less |
| | 2 | 3-8 weeks |
| | 3 | 2-6 months |
| | 4 | 6-12 months |
| | 5 | More than 12 months |
| IN (incidence) | 1 | Unusual |
| | 2 | Occasional |
| | 3 | Common |
| | 4 | Very Common |
| | 5 | Most Frequent |

Energy is dissipated when an injury is produced. In AIS, injuries have been ranked by the amount of energy dissipated in their production, which considered in ED criteria. Injuries of equal or similar ED may be vastly different in their threat to life, such as fracture in femur versus brain injury. TL is being considered. PI, commonly those of the musculoskeletal system, impair a patient's ability to work or perform activities of daily living. TP identifies those injuries, which require long treatment periods. Injury diagnoses were ranked according to the frequency with which they occurred, based on the clinical experience of the treating physician, and this is considered in criteria. Table 2-23 shows the AIS injury scaling of respective criteria in quantitation terminology [198].

## 2.12.2 HEAD INJURIES

While looking specifically at statistics of head injuries, Otte and Felten [199] found that fractures of the lower jaw and skill based occurred only at speed over 30 km/h. Severe brain injuries of severity AIS[*] 3 were sustained at relative speed as low as 11 km/h. Injury to the middle facial skull occurs only in the higher speed ranges of over 40km/h. For impact where contact was oblique from below the head and the lower jaw was fractured without injury to the bony middle facial skull, the minimum relative speed was estimated to be 35 km/h, whereas for a lower jaw fracture with middle-face fractures a minimum speed of 45 km/h is necessary. Some typical head injuries are shown in the following of this section.

*Skull fracture* – skull fracture can occur with or without brain damage, but is in itself not an important cause of neurological injury [200]. Skull fracture can be either open or closed. A closed fracture is a break in the bone, but with no break in the overlying skin. An open fracture, on the other hand, is a contiguous break in both the skin and underlying bone. An open fracture is more serious due to the accompanying risk for infection.

*Cerebral Contusion* – cerebral contusions are bruises of the brain caused by haemorrhages of small blood vessels. Contusions are the most frequently found type of head injury, crest of gyri being common places for them to occur. In these cases, the contusions are wedge-shaped with apex extending into the white matter [201]. Contusions are most often multiple and are frequently associated with other lesions such as cerebral haemorrhage, subdural haematoma and extradural haematoma. Clinical findings indicate that most cerebral contusions occurs at the frontal and temporal lobes [202], regardless of whether the patients had experienced frontal or occipital impacts [203]. This suggests that the inner geometry of the skull may contribute to contusions. Observations in patients and in experimental subhuman primates have shown that severe and fatal damage to the brain resulting from head injury can be sustained without visible

[*] Abbreviated Injury Scale – refer previous section

contusions. Furthermore, a good recovery is usual, despite having sustained severe cerebral contusions [203].

*Intracranial Haematoma* – Intracranial haematoma is an accumulation of blood inside the head caused by a haemorrhage. The colour of the haematome indicates the kind of blood vessel that has been ruptured (arterial blood is bright red and venous blood is dark red). Bleeding inside the head may not be the main cause of injury, as a result of the bleeding the haematoma may expand causing compression and shifting of the brain and an increase in intracranial pressure. This expansion of the haematoma determines the amount and severity of the resulting neurological injury.

*Cerebral Concussion* – in a layman term, cerebral concussion is impairment of consciousness, decrease in responsiveness and posttraumatic amnesia. Different researchers give their own interpretations and revisions of others [204, 205, 206].

### 2.12.3 HEAD INJURY MECHANISM

An impact to the head results in acceleration or deceleration of the head, which leads to inertia loading of the intracranial structures. Two main types of acceleration contribute to the head injury. They are:

- Translational acceleration – the head accelerates along a path without rotation of the head
- Rotational acceleration – the head undergoes an angular movement about the centre of rotation, around the centre of gravity of the head for the case.

Combination of both rotational and translational accelerations can happen. Lee *et al.* [207] pointed out that the centripetal component is usually negligible compared to the tangential translational acceleration. In reality, combination of both types of accelerations often occurs in traffic accidents, where a non-centroidal rotational acceleration around the neck – head joint.

McElhaney *et al.* [208] classified two distinct types of loading during the crash impact, they are:

- An impact or blow involving a collision of the head with another solid object at an appreciable velocity. This situation is generally characterised by large linear (translational) accelerations and small angular accelerations during the impact phase.
- An impulsive loading including a sudden head motion without direct contact. The load is generally transmitted through the head-neck junction upon sudden changes in the motion of the torso and is associated with large angular acceleration of the head.

Reference [209] shows that in the moderate but survivable automotive crash environment (30 mps barrier equivalent) no significant head injuries occur when the fully-belted occupant rides the crash down without any direct impact at the head. The second type of loading, impulsive without contact, can only cause injury in more severe crashes. Mackay *et al.* [210] also pointed that no serious head or neck injury has been seen without direct contact.

### 2.12.4 HEAD INJURY CRITERIA

In order to properly design devices aimed at minimising head injury in the automotive crash environment, engineers require a means of predicting potential injury [208]. The automotive crash environment encompasses a wide range of impulse duration and directions. Thus a viable injury criteria must take into the account of these factors, including the above mentioned injury mechanisms.

#### A. Wayne State Tolerance Curve (WSTC)

The WSTC is considered to be the foundation of research on human head injury criteria. It was introduced by Lissner and Gurdjian [211] in 1960. The WSTC further evolved with the work in references [212, 213]. The curve was originally developed from data obtained by dropping embalmed cadaver heads onto unyielding flat surfaces. Linear skull fracture was used as the criterion of injury. The present curve gives the tolerable average acceleration in A-P (anterior – posterior) direction as function of the pulse duration. Figure 2-16 shows a re-plotted WSTC. Only translational accelerations were used in the development of the curve, which was obtained from different experiments with cadavers (2<*t*<6ms), cadavers and animals (6<*t*<10ms) and volunteers (*t*>10ms).

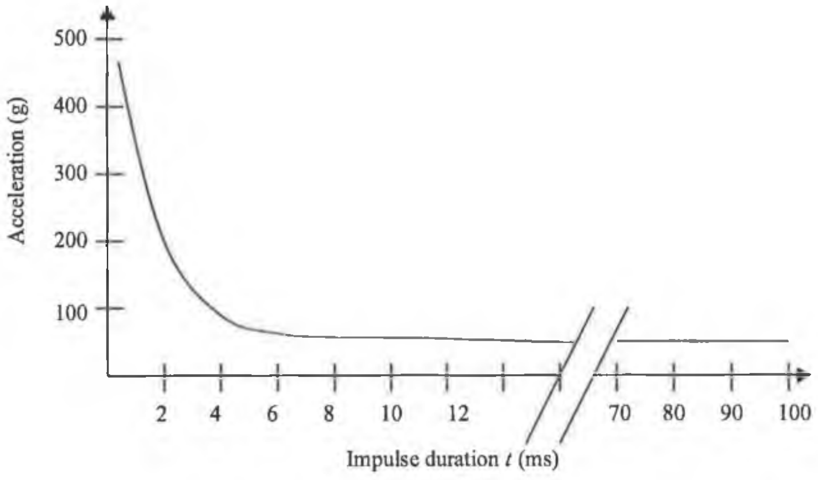


Figure 2-16: Re-plotted Wayne State Tolerance Curve

A given average acceleration at a particular pulse duration which lies below the WSTC is considered to cause at most cerebral concussion without permanent after-effects, while any point which lies above the curve is considered to be dangerous to life. McElhaney *et al.* [208] pointed the shortcoming of using WSTC. It is due to difficulty in defining the average acceleration and duration, as multiple impacts are quite common in crash-impact scenario. In spite of the many interpretative difficulties associated with WSTC, it has been the principal source for head injury tolerance information used by the automotive safety community [208].

## B. Gadd Severity Index (GSI)

Gadd [214] introduced the GSI as a generalisation of the WSTC. Gadd argued that neither the average acceleration nor the peak acceleration observed in an impact is sufficient to determine, accurately, the response of the head to an impact. The resulting injury potential is highly dependent upon the shape of the acceleration pulse. Therefore pulses with the same average acceleration but different shapes can have very different effects. To include the acceleration, the duration and the pulse shape, Gadd suggested integrating the acceleration signal over its entire duration. Gadd also suggested an exponential weighting factor (greater than 1) to the acceleration to tackle the non-linear injury potential function of acceleration magnitude. The following equation was suggested.

$$GSI = \int_0^T a^n dt \qquad (2\text{-}5)$$

where

| | |
|---|---|
| $a$ | Acceleration, g |
| $n$ | Weighting factor |
| $t$ | Time, s |
| $T$ | Impulse duration, s |

The higher the weighting factor the greater the dependence of the injury threshold on the magnitude of the acceleration as opposed to the duration. The value of 2.5 was found when comparing to WSTC, thus a value of 2.5 was suggested. And the GSI of 1000 was recommended as the threshold. If the GSI is greater than 1000, the acceleration pulse is considered to be dangerous to life. If less than 1000, the acceleration pulse is then considered not to be life threatening. The threshold value of 1000 was arbitrarily chosen but appeared to agree with the WSTC and Eiband tolerance curve [215]. However, Slatternschek and Tauffkirchen [216] reported deviation of GSI up to 55% from the WTSC and significant scientific criticism has been throwing over the use of GSI as a reliable index. Gadd [217] later suggested a threshold of 1500 for non-contact loads on the head.

## C. Versace Severity Index

Versace [218] pointed out that because the WSTC was plotted for average acceleration, any comparison to the WSTC curve should be made using the average acceleration of the pulse of interest. Therefore the following equation is suggested for injury criterion.

$$VSI = \left[ \frac{\int_0^T a dt}{T} \right]^{2.5} T \tag{2-6}$$

### D. Head Injury Criterion (HIC)

Based on the Versace's suggestion, NHTSA* proposed a new criterion scheme. This was due to the problem posed with long pulse head accelerations, when using VSI to predict head injuries. The following equation is used to find the HIC index.

$$HIC = \left[ \frac{\int_{t1}^{t2} a dt}{t2 - t1} \right]^{2.5} (t2 - t1) \tag{2-7}$$

where

| | |
|---|---|
| $a$ | Resultant acceleration at the head centre of gravity, g |
| $t1$ | An arbitrary time in the pulse |
| $t2$ | For a given $t1$, a time in the pulse which maximises the HIC |

Threshold value of 1000 is used for HIS. After much discussion over many years, $t1$ and $t2$ were defined to be any two times during the entire impact duration for which HIC is a maximum value. Hodgson and Thomas [219] suggested that the HIC interval should be less than 15 ms, even if the HIC value exceeded the threshold of 1000 over a long interval.

### 2.12.5 ROTATIONAL ACCELERATION

Previous sections show the severity or extent of head injuries caused by rotational acceleration is lesser compared to translational acceleration [207, 209]. There is no criterion related, and usually ignored by researcher. Anyhow, literature shows brain injuries can be caused by rotational acceleration. Ryan *et al.* [220] showed that, for

* National Highway Traffic Safety Administration, USA

occipital impacts death was reported to occur at linear accelerations of about 200g upwards. This was accompanied with increasing rotary acceleration from 6000 rad/s$^2$ to 15000 rad/s$^2$. Unterharnscheidt [221] performed rotational acceleration tests on 24 squirrel monkeys. The monkeys were subjected to rotational acceleration of 101k rad/s$^2$ to 386k rad/s$^2$. The results showed that, lowest rotational accelerations employed (101k – 150k rad/s$^2$) caused apparently no primary or secondary alternations in the cerebrum. However higher rotational acceleration up to 197k rad/s$^2$ caused 10 of 13 animals subarachnoid hemorrhages, combined in one instance with primary traumatic hemorrhages in the oculomotor nerve, and tears and avulsions, mainly of veins and capillaries, in superficial cortical layers in 8 animals. Accelerations of more than 200k rad/s$^2$ caused severe primary traumatic hemorrhages in the cortex and white substance. None survived with rotational accelerations of more than 300k rad/s$^2$.

### 2.12.6 LOWER LIMBS INJURY CRITERIA

No injury criterion is available for lower limb injury. As pointed from the previous section, the injuries would be on soft tissues and bone fractures, which caused by direct impact during collision.

Johnson *et al.* [222] suggested that 6.2kN is the appropriate tolerance level for the femur. The author also pointed that, traumatic fracture is not the appropriate criterion but rather post-traumatic arthritis of the knee and pelvic joint, which will only occur long after an accident. To prevent pelvis internal injury, 60g for 3ms is the threshold.

### 2.12.7 THORACIC AND ABDOMINAL INJURY CRITERIA

Thoracic and abdominal injuries are not common in motorcycle crash (around 10 percent in total, refer to Table 2-22). Results from work on volunteers and cadavers and through accident studies, rib fracture is believed to occur for distributed loads of between 5.3k N and 8.9k N. Acceleration of 60g for up to 10 ms can be used as the threshold value for the rupture of the great vessels in the thoracic cage, although it may happen for some individuals with this situation [222].

# Chapter 3: DEVELOPMENT OF FINITE DIFFERENCE MODELLER

## 3.1 INTRODUCTION

First generation of the system developed is illustrated in this chapter. A general introduction is presented through illustration of the system components and respective functions. The component hierarchy, system flow, and data flow is shown. The chapter is divided into three sections. In each section, the definition and implementation of classes, input-output file systems, and message handlers of corresponding component are explained. Discussions and summaries of each component is presented at the end of each section. The system application methodology is presented as well.

## 3.2 FDM OVERVIEW

The initial development is named as Finite Difference Modeller (FDM). The naming was given arbitrarily. FDM is the first generation of the numerical based simulation system developed in this study.

A FD code in FORTRAN*, solving a stacked ring impact problem, was studied before the development of the system. The code is condition-dedicated and problem-oriented, which is only suitable to solve a specific problem. The core FD model was based on Witmer *et al.*'s model [1], which is a general model in solving structural impact in 2D space. Thus, generalisation has been emphasised since the beginning of the development. The development was also aimed to solve motorcycle-rider crash problem. The up-to-date capabilities of the FDM are pin joint with joint resistant,

* FORmula TRANslator, a programming language

contact-impact algorithm, branching of the 1D structure (which is not allowed in the formulation of Witmer *et al.*'s model).

Microsoft Visual C++ was chosen to develop the FDM. Version 5.0 was the latest version during the initial development. FDM was compiled on Windows 95 platform. Windows user interface was incorporated as well. All the components of the FDM are incorporated under one system with multiple documents – multiple views interface setup. Object Oriented Programming (OOP) technique is implemented. The FDM consists of 10 distinct classes. OOP is used for handling the interface of the Windows and handling multiple documents and multiple views. Only partial use of the OOP was considered throughout the implementation, although the OOP-like structured variables are widely used in the FDM. This was due to the influence of the previous "reference", the stacked-ring codes, which was written with structural programming method in FORTRAN.

The latest version of the FDM is set to handle up to 150 nodes and 25 links. This takes up to 13000 long double variables. Each long double variable is capable of handling 19 decimal points with exponential power more than ±300. These boundary limits can be set higher, with the risk of memory handling problem. Dynamic memory is not allocated in this system. Allocation of dynamic memory will solve the overloaded memory problem, but will lead to a much slower solving mechanism.

## 3.3 FDM COMPONENTS AND FUNCTIONS OVERVIEW

FDM consists of three major components. They are FDM pre-, post- and the processor itself. The three components are physically embedded in a single system.

The main function of the pre-processor is to create an environment for the user to build the required model. The present pre-processor version of the FDM is relying on the user-input file. In other word, the pre-processor is a viewer of the input data file. All amendments have to be made on the data file with a text editor. Validation of the model can be made with the pre-processor. The pre-processor helps generating links and nodes from the model input file. This process is analogy to the meshing process.

After extracting the data file with the pre-processor, the data is ready to be sent to the processor after the meshing task is finished. The FD model of Witmer *et al.* [1] is implemented in the FDM processor. Initialisation of the initial conditions is a must. The processor will process the model conditions at every time step. The process iterates with information yield from previous time step till a predetermined maximum iteration number. The result of the processor is stored in the output data file for further processing. The output data are in binary form, which make the reading and writing process faster and more efficient.

The post-processor will retrieve the processed data from the FDM processor. The main function of the post-processor is to further interpret the data in a more presentable way, according to the user requirement. The deformation and transient motion are animated. Additional information, such as the respective bending moment or shear stress, can be shown in conjunction with the displacement location of each link and node. Extra output text file can be generated, in which the text-based data are sorted in a legible form. The text file can be imported by any commercial packages like Microsoft Excel for further interpretation, such as generating graphs.

## 3.4 FDM COMPONENT HIERARCHY AND SYSTEM FLOW

The component hierarchy is designed in such a way to cope with the requirement of the system flow. The following diagrams are being generated to help describing the FDM component hierarchy and system flow respectively.

In general, the user chooses to use the pre- or post-processor after entering the FDM interface, indirectly. No physical dialog box or rigid selection is necessary for the user at the initialisation of the FDM. The user opens the input file, which can be either the pre-processor input file or the post-processor input file. The system will automatically determine proper executions. The FDM processor sits under the pre-processor in the system hierarchy. This means, the pre-processor has to be invoked before any process to be executed by the processor. As with multiple documents interface (MDI), a user can open and view several files at the same time. Furthermore, the FDM enables opening and viewing files of different file type simultaneously. Figure 3-1 and Figure 3-2 show the component hierarchy chart and the system flow chart respectively.

## 3.5 FDM DATA FLOW

General data flow of the FDM can be described with Figure 3-3. The figure also shows the flow of general states (which defines the pre-, post- and the processor itself). Three types of data file are involved in the FDM, which are the FDM input text file, the FDM processed binary data file and the FDM processed text data file. As mentioned earlier, the system can detect the input file type by referring to the file extension and invokes appropriate components. When a particular component is invoked, the system changes to the corresponding state.

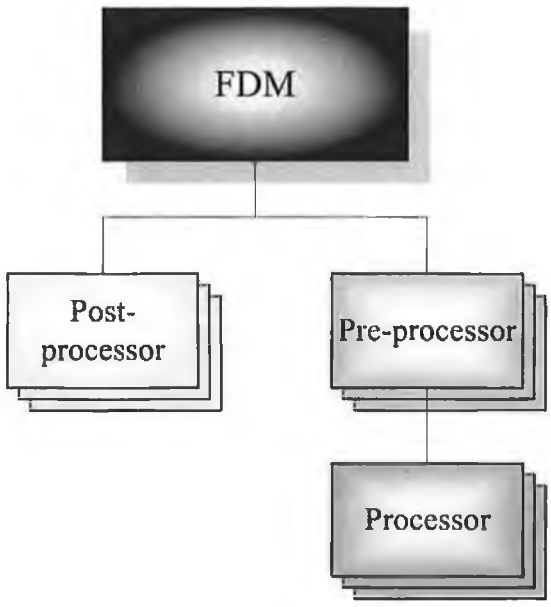


Figure 3-1: Component hierarchy of FDM

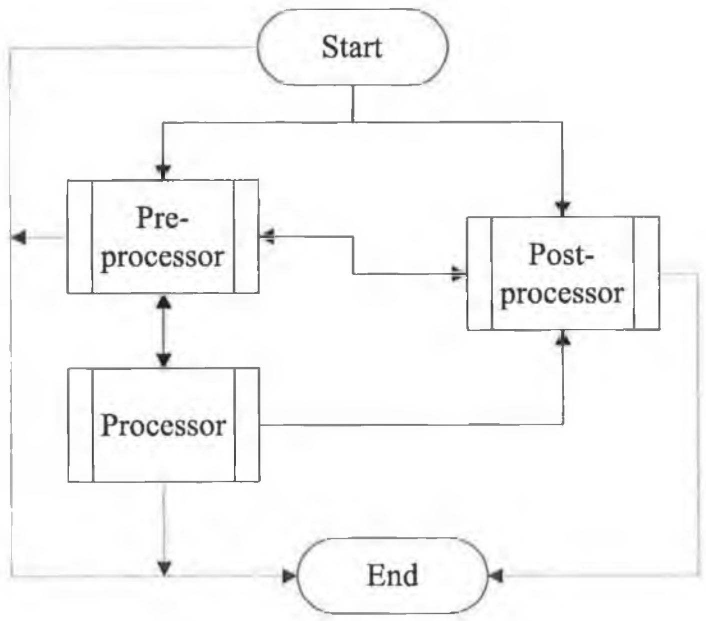


Figure 3-2: System Flow of FDM

In a pre-processor state, the data from the pre-processor file will be read in the pre-processor document class. The data is used by the pre-processor view class as well as to display onto the screen. The document class is shared among the pre-processor and the processor itself. It consists of both processors' functions and procedure, which includes meshing (node-link generation), and FD simulation algorithms. In the figure, the state of the processor in fact is "a state to process data" symbols in order to differentiate the pre-processor and the processor data flow. Physically, it should be included in the document class. The revisit of the processed interval data at the "Processor" state shows that the processing of the FDM is based on the previous time step data to predict the response. For a pre-defined time step, the data will be sent to the view class in order to be shown on the screen. Besides that, the data are also being sent to the FDM output binary file and FDM output text file (optional). The binary data file can be used by the post-processor, while other packages can use the text-based data file for further interpretation of the data.

When the post-processor is invoked, the data from the FDM output binary file is serialised in the post-processor document class. Different options can be made from animating the motion to showing particular time frame motion. Different post-processor algorithms are invoked to process the data accordingly. The processed data are sent to the view class for display on the screen.

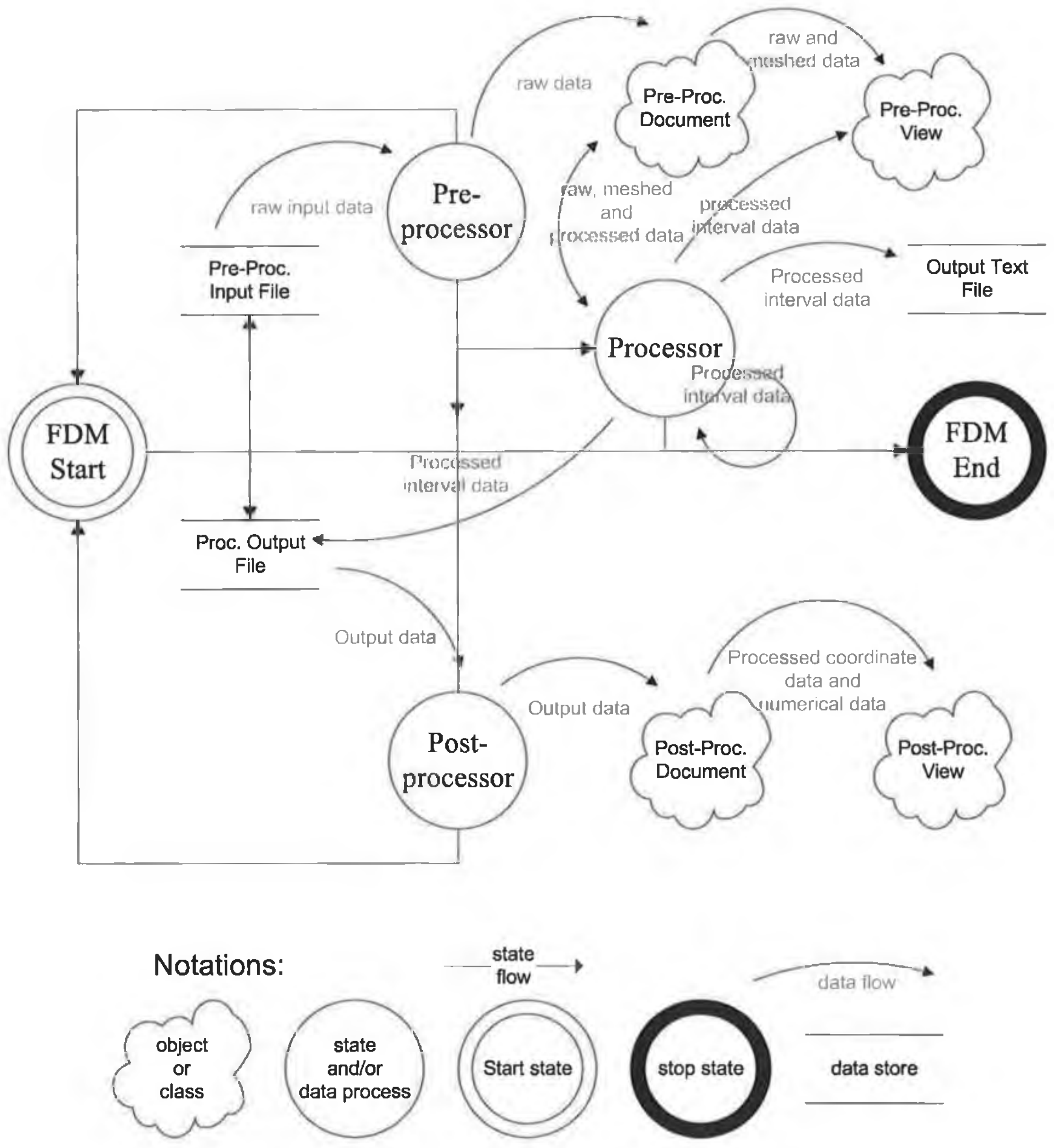


Figure 3-3: Data flow of FDM

## 3.6 FDM CLASSES OVERVIEW AND INITIALISATIONS

FDM only consists of 10 classes. Three of them are dealing with dialog boxes for user interface. Two document classes and two view classes for the pre- and post- processors respectively. Two others are dealing with Multiple Documents Interface (MDI) frames. The last one is for the main initialisation of the application, such as creation of the Windows, and binding of the menus, documents and views.

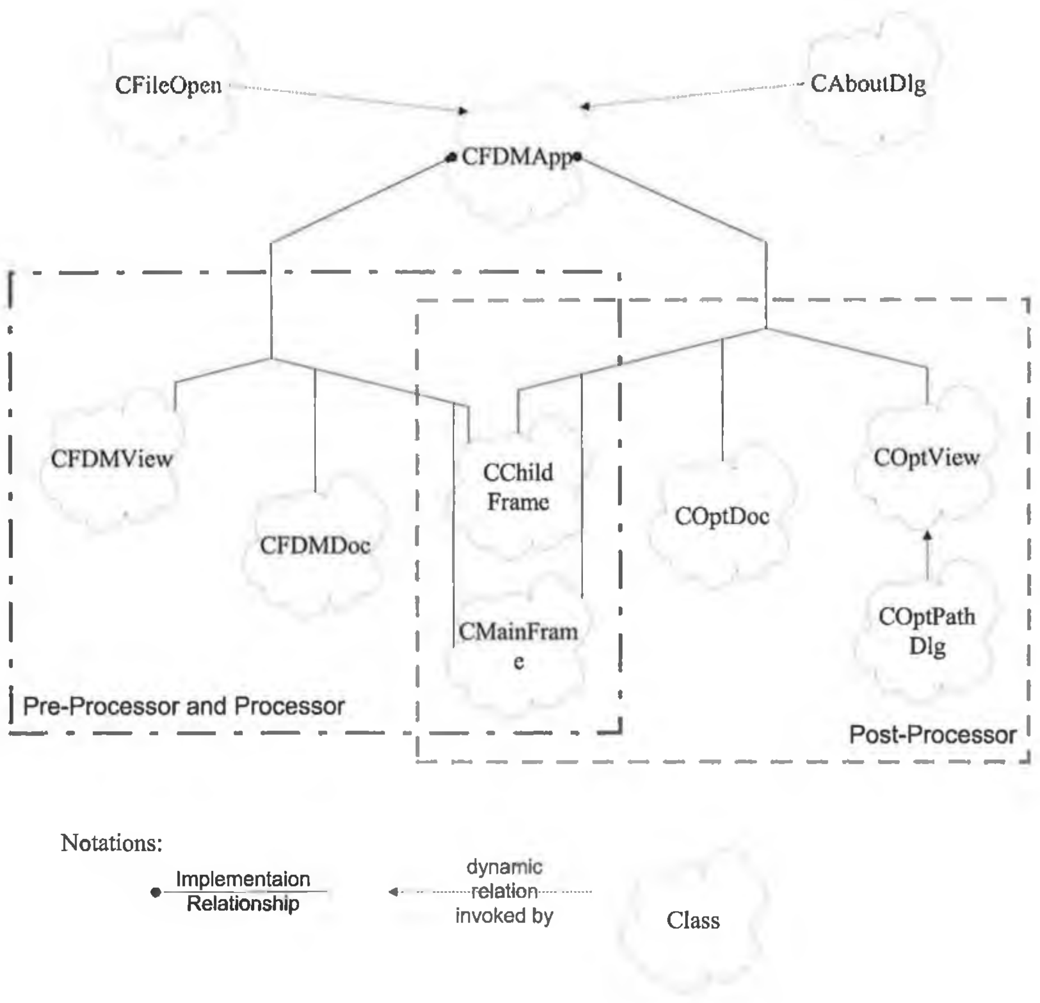


Figure 3-4: FDM classes organisation

Figure 3-4 shows the relationships of respective classes. The CFDMApp is a public derived class of CWinAPP and acts as the starting point of the FDM. Initialisation is made in the implementation of the CFDMApp class. The significant initialisation can be placed in CFDMApp constructor function, CFDMApp::CFDMApp(), but not advisable. Initialisation codes should be placed in the CFDMApp::InitInstance(). No global variable is involved in FDM, thus no initialisation is required. Besides the variable initialisation, the FDM standard profile is invoked, where the previous or history information was stored. The information mentioned consists of the file type registration, and history file list (MRU). The relationships among the CFDMDoc, CFDMView and CChildFrame, and the relationships among the COptDoc, COptView and CchildFrame are defined in the CFDMApp::InitInstance(). CFDMDoc, and CFDMView are the pre-processor and processor document class, and view class. While, COptDoc and CoptView are the document class and view class for the post-processor. The corresponding menu resource ID* has to be specified as well during the binding of the above mentioned relationships. A resource ID is a pointer to the respective resource location. The resource is referring to menu, icon, accelerator table, string table, dialog or toolbar in general. The resources are defined during the development and stored in FDM.rc, a corresponding resource file of FDM. In this case, three separate menu resource IDs are used, which are for the pre-processor and the processor, the post-processor, and at initial state where the user has not opened any data file (may it be FDM pre-processor input file or FDM binary output data file). The IDs also include file extension information, which will be used to identify the proper binding to be used for different file type.

Both the pre- and post- processors share CChildFrame class and CMainFrame class. The CChildFrame stores information about the implementation of Multiple Documents Interface (MDI) child frames. While the CMainFrame contains the implementation of the Main Frame, such as the toolbar, which is defined under the CFDMApp::InitInstance() as well. As a matter of fact, the CMainFrame is common throughout the whole FDM, including the initial state of FDM.

* IDentity number, usually an integer value

The CFileOpen is invoked when a user sends a *file open* request. The request is sent through the menu, *File – Open Input File…* (The respective menu items and functions are explained in the following section). An instance based on CFileOpen is created, where the file type specification is made. An instance of CAboutDlg class is created when *Help – About FDM…* request is sent. The class controls the "About" dialog box, which shows some information about the FDM.

## 3.7 FDM MAINFRAME MENU SYSTEM

The menu system for the initial state is called MainFrame, where the FDM *Start* state is. The menu resources coding can be found under section A.4.1 of Appendix A. The main menu consists of three menus. They are called *File*, *View*, and *Help*. The functions and hierarchy structure of the MainFrame menu system are shown below:

1. File
    - 1.1. New – generate new input data file. Initial design was to embed a text editor, such as notepad, to give the user a space to create and/or edit the input data file.
    - 1.2. Open Input File… – user uses this sub-menu to open data file, may it be the pre-processor input data file or the processor binary output data file (the input file of the post-processor).
    - 1.3. Print Setup – to set the printing properties. Standard print setup dialog box is used.
    - 1.4. Recent File – where to display the previous active files. The number of files in the list is set to 4 by default.
    - 1.5. Exit – close all the windows and exit from the FDM.
2. View
    - 2.1. Toolbar – a Boolean selector, whether to show toolbar or hide it.
    - 2.2. Status Bar – a Boolean selector, whether to show the status bar or hide it.
3. Help
    - 3.1. Help Topics – invoke the help mechanism, where help can be found. Only the standard help functions are available.

3.2. About FDM – display the About dialog box.

## 3.8 HANDLING OF *ID_FILE_OPEN* COMMAND MESSAGE

With reference to Coding A-2, the *File-Open Input File…* is represented by an ID namely *ID_File_Open*. A command message of this ID is invoked when a user selects *File* in the main menu and selects *Open Input File…* in the sub-menu. The command message is tracked by the CFDMApp class and handled by its member function, called OnFileOpen(). The following Nassi-Schneiderman [223] chart shows the procedure flow of the CFDMApp::OnFileOpen().

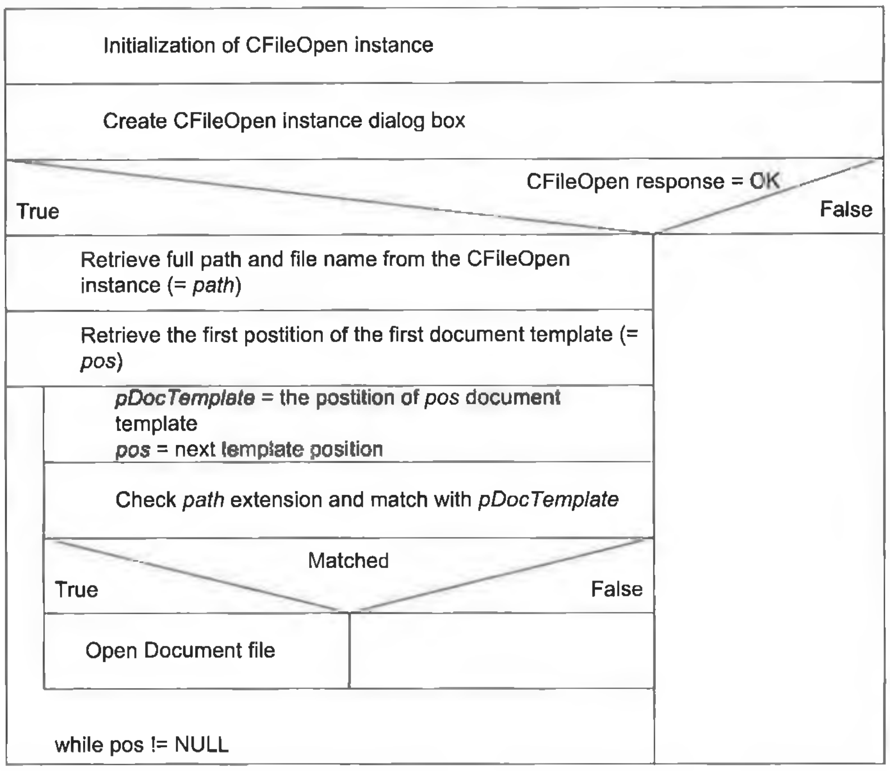

Initialization of CFileOpen instance
Create CFileOpen instance dialog box
CFileOpen response = OK
True
False
Retrieve full path and file name from the CFileOpen instance (= *path*)
Retrieve the first postition of the first document template (= *pos*)
*pDocTemplate* = the postition of *pos* document template
*pos* = next template position
Check *path* extension and match with *pDocTemplate*
Matched
True
False
Open Document file
while pos != NULL

Figure 3-5: Nassi-Schneiderman chart of ID_File_Open command handler

An instance based on CFileOpen class will be initiated. CFileOpen instance will limit the file extension, to be shown and to be selected, to two type, which are *inp* and *opt* for pre-processor input data file and processor output binary data file respectively. A modal dialog box* of CFileOpen instance is called. Upon the termination of the dialog box, a user can press *ok* or *cancel* button.

The handler searches through all the document templates to match the extension, which has been specified initially in the CFDMApp::InitInstance(). If found, the file is opened with the appropriate combination of document, view and childframe classes. Or else an error message is shown, and no action is taken for the selected file.

## 3.9 FDM PRE-PROCESSOR

As mention in the previous section, the FDM Pre-processor and the FDM Processor sit on the same platform, which are handled by the CFDMDoc and CFDMView class. Thus, only the pre-processor part of both classes will be mentioned under this section and its sub-sections.

The main function of the pre-processor is to help the user to develop and generate the numerical model, which will be processed by the processor at a later time. The FDM pre-processor is a simple and semi-automated pre-processor. The FDM pre-processor reads the data from the pre-processor input file and stores in the CFDMDoc member variables. The CFDMView class uses the information stored to display on the screen. Meshing command message is fired with a user request, which prepare the necessary information and initialisation to be processed by the FDM processor.

### 3.9.1 MENU DEFINITIONS

Coding A-3 of Appendix A shows the menu resource code for the Pre-Processor. The main menu of the FDM Pre-processor consists of six popup menus. The main functions of the pre-processors are gathered under *Run FDM* popup. They are:

* A modal dialog box closes automatically when the user presses the OK or Cancel buttons or when the code calls the *EndDialog* member function.

- Node & Initial Cond. – meshing process of the model, where nodes and links are generated, and initial conditions are assigned.
- FDM – execute the FDM Processor mechanism.

### 3.9.2 GENERAL CLASS INTERACTIONS

General class interactions are shown in the Figure 3-6 and discussed in this section. FDMDoc class serialised the data with the pre-processor input data file. The serialisation is unidirectional – retrieving. The data is then displayed on the screen by CFDMView class. Viewing related functions and procedure are stored in both CFDMDoc and CFDMView classes. Different command messages are handled by different member functions of different classes. The CFDMDoc class may issue messages to FDMView to perform certain tasks such as updating the screen display.

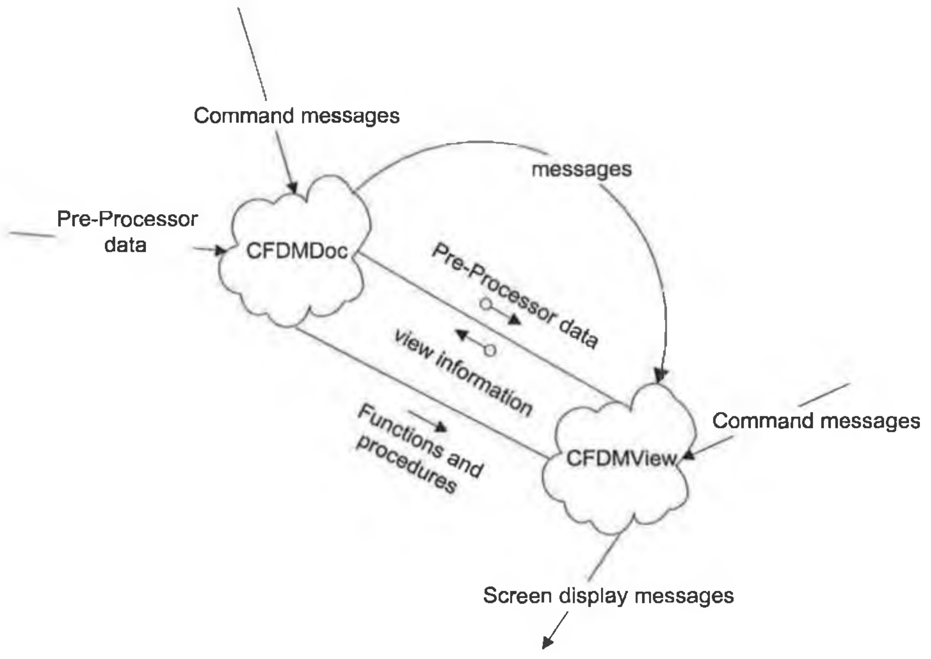


Figure 3-6: Pre-processor classes interactions

### 3.9.3 CFDMDOC CLASS

The CFDMDoc class is a public inheritance (unrestricted transferability) of the CDocument class. The main functions of the CFDMDoc is to store the information of the corresponding document and implement any changes or processing according to the user request (through command messages). This class consists of six structure types, 23 attributes (member variables), 10 implementation functions (member functions), which excluding the constructor and destructor functions, two overriding functions and handling two command messages. All the members mentioned above which may only be used by either the pre-processor and/or the processor alone.

CFDMDoc overrides two member functions of its ancestors, namely *Serialize(Carchive& ar)* of *CObject* and *OnNewDocument()* of *CDocument. Serialize* function will be invoked when the document is storing or retrieving data to or from the physical storage. While *OnNewDocument()* is executed when a new document is created. CFDMDoc handles two messages, they are command message on *ID_RUN_NODEINIT*, and menu item update message on *ID_FDM*

### 3.9.4 CFDMVIEW CLASS

The CView class is a public base class of the CFDMView. The main function of this class is to handle display on the screen. In addition, the implementation coding such as message handling, can be incorporated as well.

Only two member variables in the CFDMView class, namely *m_ClinetX* and *m_ClientY*. Both of them are integer. They store the co-ordinate of the client windows, which is used to set the extents of window and viewport associated with the device context*.

* A device context is a Windows object that contains information about the drawing attributes of a device such as a display or a printer.

The CFDMView handles one command message, *ID_FDM*, and one window message, *On_WM_SIZE()*. Furthermore, three public virtual functions and four protected virtual functions are overridden.

The public functions are for displaying information on the screen, initialisation before creating the document window, and preparing or initialising the device context. For the protected functions, they are called while a print command message is sent.

### 3.9.5 INPUT FILE SYSTEM

One input file system, namely the FDM pre-processor input data file, is involved in pre-processing. The output of the pre-processor is stored in the memory, and is ready to be used by the processor. The input data file describes the model and some necessary information for the simulation, such as the screen display, the time step size, and etc. Detail description of the file system can be found under section A.2 of Appendix A.

### 3.9.6 MESSAGE HANDLING

Messages are handled with one or several implementations (member functions) of the CFDMDoc and CFDMView classes. In the following sub-sections, the handling of different important messages are discussed and algorithms are explained. Only important implementations are covered.

#### A. Serialization Message

When a user wants to save or read a file to or from the physical storage (for example, hard disk), the *Serialization* message is fired. The *Serialize* member function of CObject class is handling this message by default. Overriding in the CFDMDoc class of this function is necessary in order to customise the handling of the input-output of the data.

## A.I Message and Data Flow

The message flow is very straightforward, the message fired and the CFDMDoc::Serialize function handles it and finish the input-output tasks with CArchive instance, *ar*. Figure 3-7 shows the message and data flow diagram of *Serialization*.

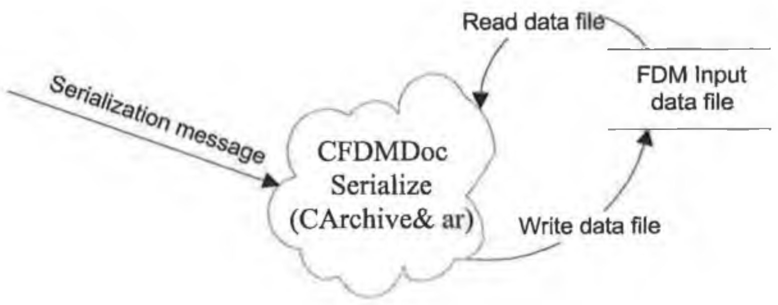


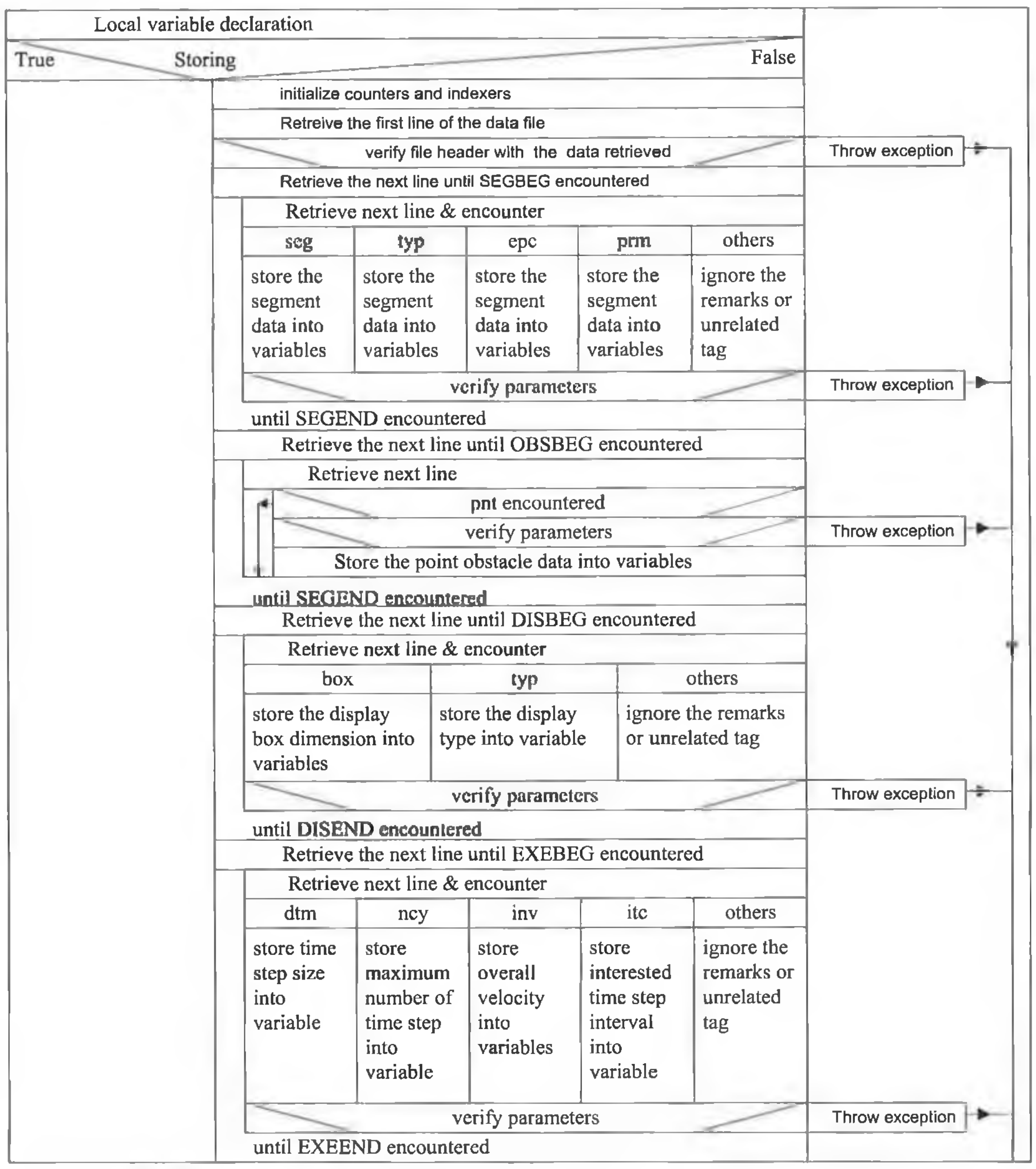
Figure 3-7: Message and data flow of Serialization

## A.II Implementations

The CFDMDoc::Serialize function only retrieve data from the FDM pre-processor data file. No storing operation is defined in the function. The *ar* is an instance of CArchive class, which brings forward with the *Serialization* message to the CFDMDoc::Serialize function. The *ar* provides the storing and retrieving stream of the document. The serialization function has the syntax below:

**CFDMDoc::Serialize** (CArchive& *ar*)

Figure 3-8 shows a Nassi-Schneiderman chart of the CFDMDoc::Serialize implementation. While processing the input data file, exception will be through if any parameter mismatch occurs. Exceptions are catch, error message is displayed and no further retrieving activity is carried out after an exception being fired. The status of the FDM is changed to "Reading Data File". The display will automatically updated and the CFDMView::OnDraw is called.



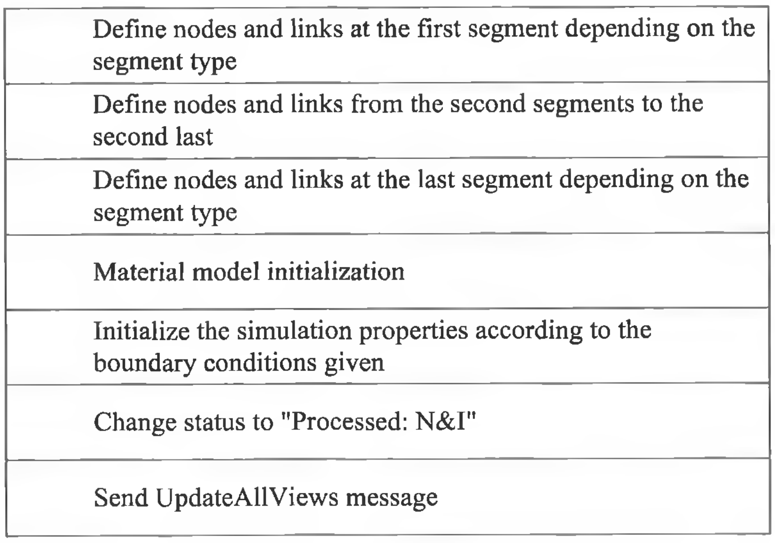
Local variable declaration
True
Storing
False
initialize counters and indexers
Retreive the first line of the data file
verify file header with the data retrieved
Throw exception
Retrieve the next line until SEGBEG encountered
Retrieve next line & encounter
seg
typ
epc
prm
others
store the segment data into variables
store the segment data into variables
store the segment data into variables
store the segment data into variables
ignore the remarks or unrelated tag
verify parameters
Throw exception
until SEGEND encountered
Retrieve the next line until OBSBEG encountered
Retrieve next line
pnt encountered
verify parameters
Throw exception
Store the point obstacle data into variables
until SEGEND encountered
Retrieve the next line until DISBEG encountered
Retrieve next line & encounter
box
typ
others
store the display box dimension into variables
store the display type into variable
ignore the remarks or unrelated tag
verify parameters
Throw exception
until DISEND encountered
Retrieve the next line until EXEBEG encountered
Retrieve next line & encounter
dtm
ncy
inv
itc
others
store time step size into variable
store maximum number of time step into variable
store overall velocity into variables
store interested time step interval into variable
ignore the remarks or unrelated tag
verify parameters
Throw exception
until EXEEND encountered

Figure 3-8: Nassi-Schneiderman chart of CFDMDoc::Serialize implementation

## B. Node-Link Generation Command Message

The ID for node-link generation command message is *ID_RUN_NODEINIT*. Activation of the message is by pressing the menu item (please refer Coding A-3). At the final stage of the implementation, an update screen display message is sent to the

CFDMView class. The following shows the Nassi-Schneiderman chart of the implementation in the CFDMDoc::OnRunInit().



Define nodes and links at the first segment depending on the segment type

Define nodes and links from the second segments to the second last

Define nodes and links at the last segment depending on the segment type

Material model initialization

Initialize the simulation properties according to the boundary conditions given

Change status to "Processed: N&I"

Send UpdateAllViews message

Figure 3-9: Nassi-Schneiderman chart of CFDMDoc::OnRunInit implementation

## C. Window Display Update Message

Every time a window display required an update, the overriding function CFDMView::onDraw will be called. This message is fired whenever there is any change on the screen such as overlapping the window with other application window(s), resizing the windows, et cetera. Any CFDMView and CFDMDoc class member function can fire the message. The message carries a parameter called pDC, a pointer to the device context class (CDC). Figure 3-10 shows the Nassi-Schneiderman chat of the implementation in the CFDMView::OnDraw. The drawing device context is customised under the CFDMView::OnPrepareDC, where the extents and the viewport of the window are set, in order to select the scale and orientation of the co-ordinate system used automatically. The setting is depending on the co-ordinates provided in the input data file defining the display box.

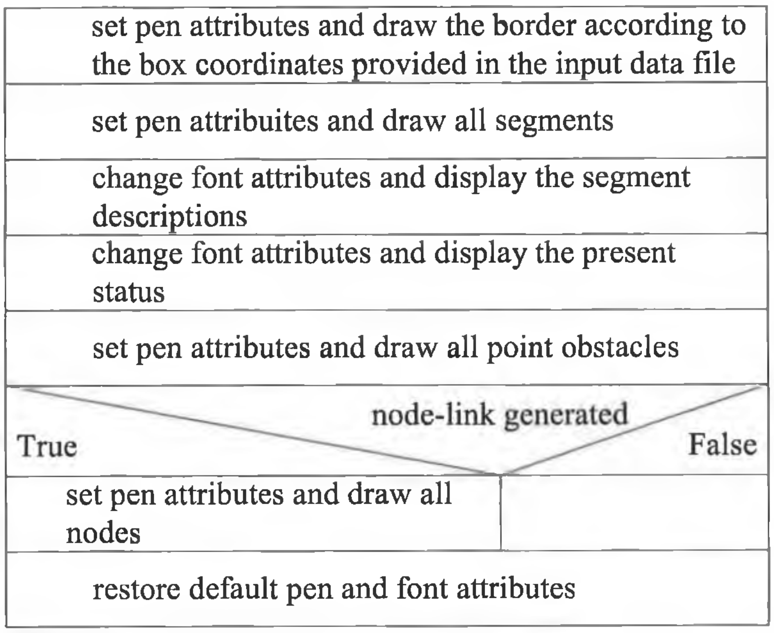

set pen attributes and draw the border according to the box coordinates provided in the input data file
set pen attribuites and draw all segments
change font attributes and display the segment descriptions
change font attributes and display the present status
set pen attributes and draw all point obstacles
node-link generated
True
False
set pen attributes and draw all nodes
restore default pen and font attributes

Figure 3-10: Nassi-Schneiderman chart of CFDMDoc::OnDraw implementation

### 3.9.7 MODELLING METHODOLOGY & EXAMPLE

A user needs a text editor to create the FDM pre-processor input data file. The text editor can either be the Microsoft Windows' Notepad, or other similar application. An example of the input data file can be found in Coding A-1. The FDM pre-processor input data file must have an extension of "inp", in order to be recognized by the FDM. All the necessary information has to be provided.

The file is then ready to be accessed by the FDM pre-processor. Figure 3-11 shows the window display after reading the input data file example of Coding A-1. It is a model of human body with head, trunk and lower extermities. The human body is travelling at a speed of 40 m/s and hitting a rigid fixed point obstacle.

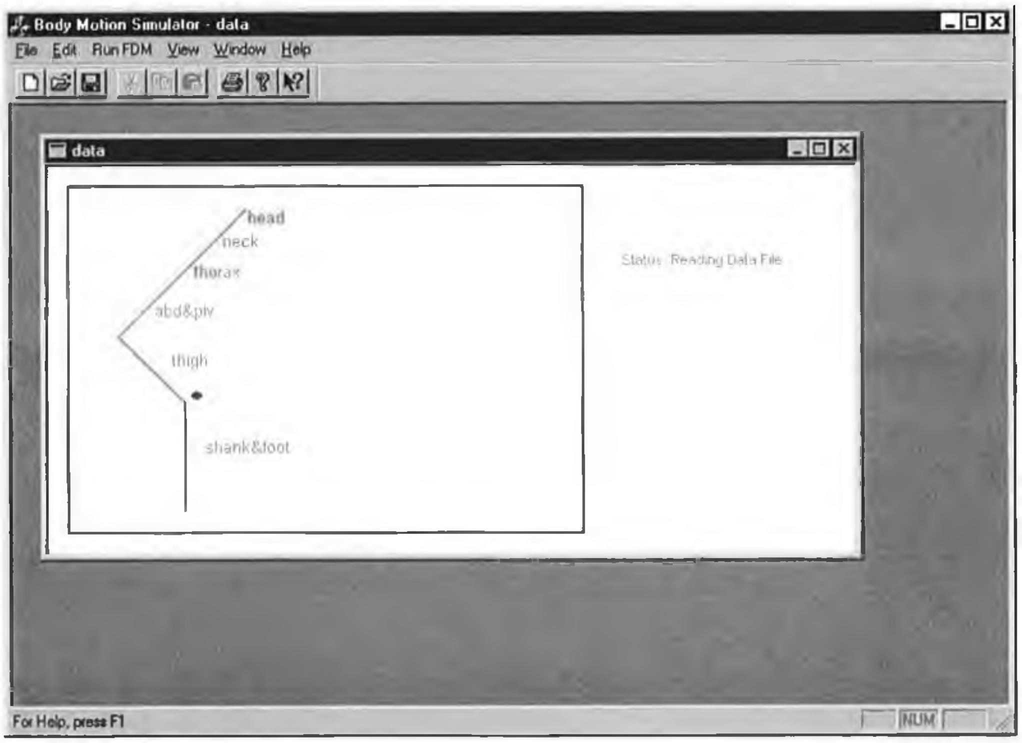

Body Motion Simulator - data
File Edit Run FDM View Window Help
data
head
neck
thorax
abd&plv
thigh
shank&foot
Status: Reading Data File
For Help, press F1
NUM

Figure 3-11: FDM pre-processor window display

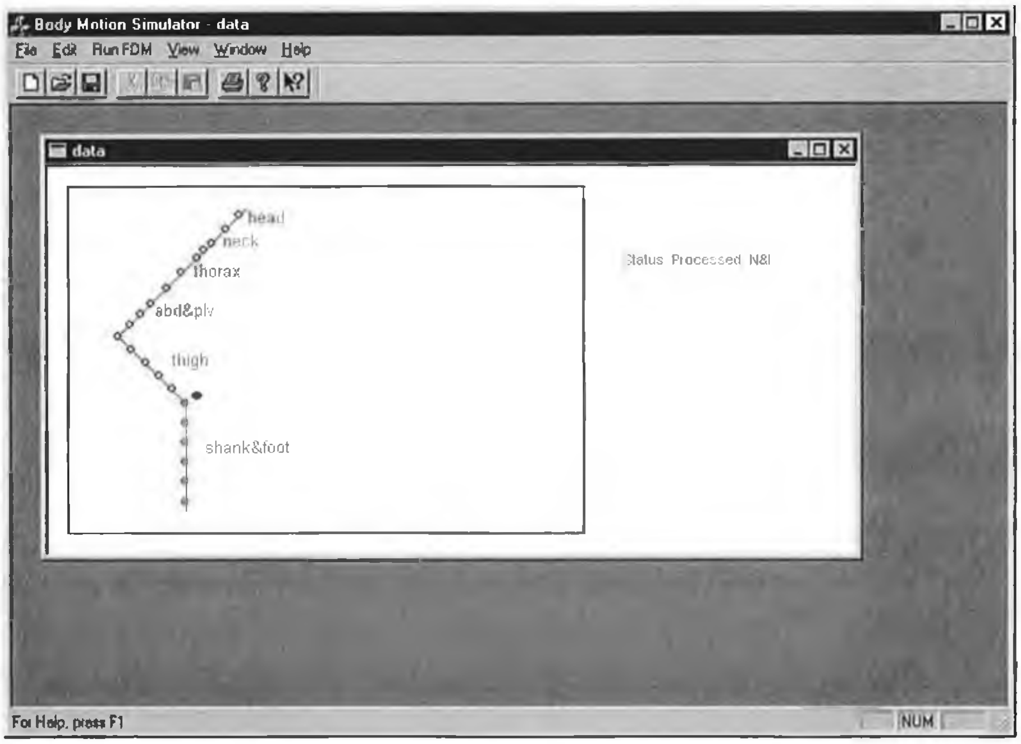


Figure 3-12: FDM pre-processor after node-link generation window display

After opening the data file, view and verify the model. Make any amendment and changes of the input data file with the text editor. Repeat until a statisfactory model achieved. Select *Run FDM – Node & Initial Cond.* in the main menu. The nodes and links will be generated and shown on the screen. Figure 3-12 shows the "meshed" version of the example coding. Validation has to be carried out at this stage. Correction of the coding has to be made with the text-editor. After the node-link generation, the model is readied to be process by the FDM processor. Figure 3-13 shows the flow chart of the modelling methodology with FDM The red dot is the location of the point obstacle with its effective radius.

### 3.9.8 DISCUSSIONS & SUMMARIES

FDM pre-processor has limited features. It served as a simple preliminary pre-processor. The pre-processor provides an interface for the user to view and validate the model created using any separate text editor.

The meshing (node-link generating) algorithm is capable to handle the four types of end points meshing option. The node-link generation is somehow limited by the modelling constraints imposed by the Witmer *et al.*'s model. Thus, breakthrough in user-friendliness is complicated.

The FDM pre-processor input data file suffers from it flexibility as well. The file system can be treated as its first version. Afford has not been put in enhancing the capability of handling boundary conditions and initial loading. The input data file can only describe the most primitive boundary conditions and initial loading. Anyhow, it is more than enough to handle simple general problems, such as the one given in the example.

Only contact-impact between the structure with fixed point is available. The bottleneck is with the FDM processor, as the author has only developed the contact-impact of such.

As a whole, the FDM pre-processor and the data file system still leave a wide space to extent and improve. The development of the FDM pre-processor halted at this initial stage, owing to the author's decision to start the development of second generation simulation system.

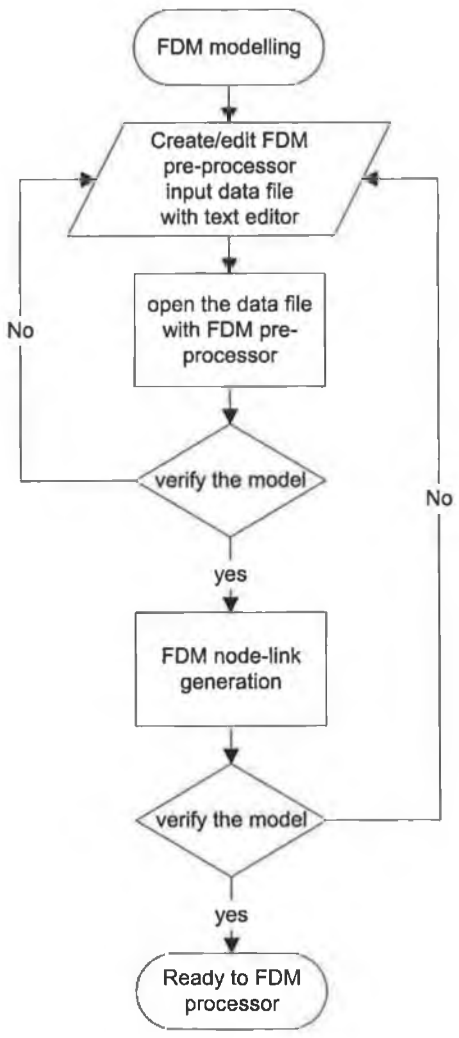

FDM modelling
Create/edit FDM pre-processor input data file with text editor
open the data file with FDM pre-processor
No
verify the model
yes
FDM node-link generation
verify the model
No
yes
Ready to FDM processor

Figure 3-13: FDM pre-processor modelling methodology

## 3.10 FDM PROCESSOR

The FDM processor's procedures are sit under the CFDMDoc and CFDMView classes. The processing starts when a user fires the *ID_FDM* command message, which can be done by selecting *Run FDM – FDM* in the main menu. The menu item will only be active after node-link being generated successfully. The following shows the message and data flow diagram.

To show the implementation of the processor, a Nassi-Schneiderman chart is shown in Figure 3-15. The theories behind the implementation can be found in Chapter 5.

### 3.10.1 OUTPUT DATA FILES

Two output files are generated in the FDM processor. The first one is a binary file, which the data is stored in binary form (in fact is hexadecimal), namely Post-processor binary data file. It can be read by the FDM post-processor as input data file. The data file will have the extension of "otp" automatically with the same file name as the pre-processor input data file. The other type is text-based data file. The file can be viewed by any text editor, or imported to any spreadsheet application, such as Microsoft Excel, for further interpretation / processing of the data. The data file will have the same file name as the pre-processor input data file, but different extension, "dta". This section only reviews the later output data file. The first type will be covered in the FDM post-processor section.

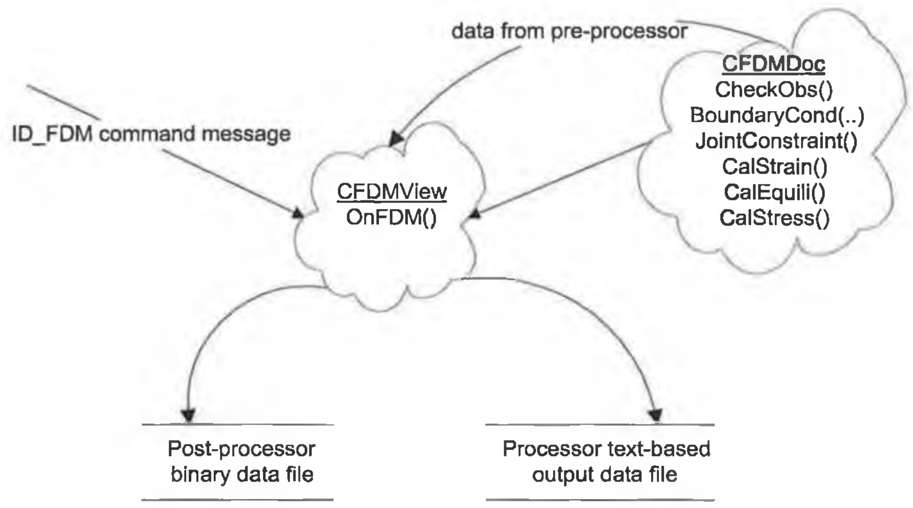
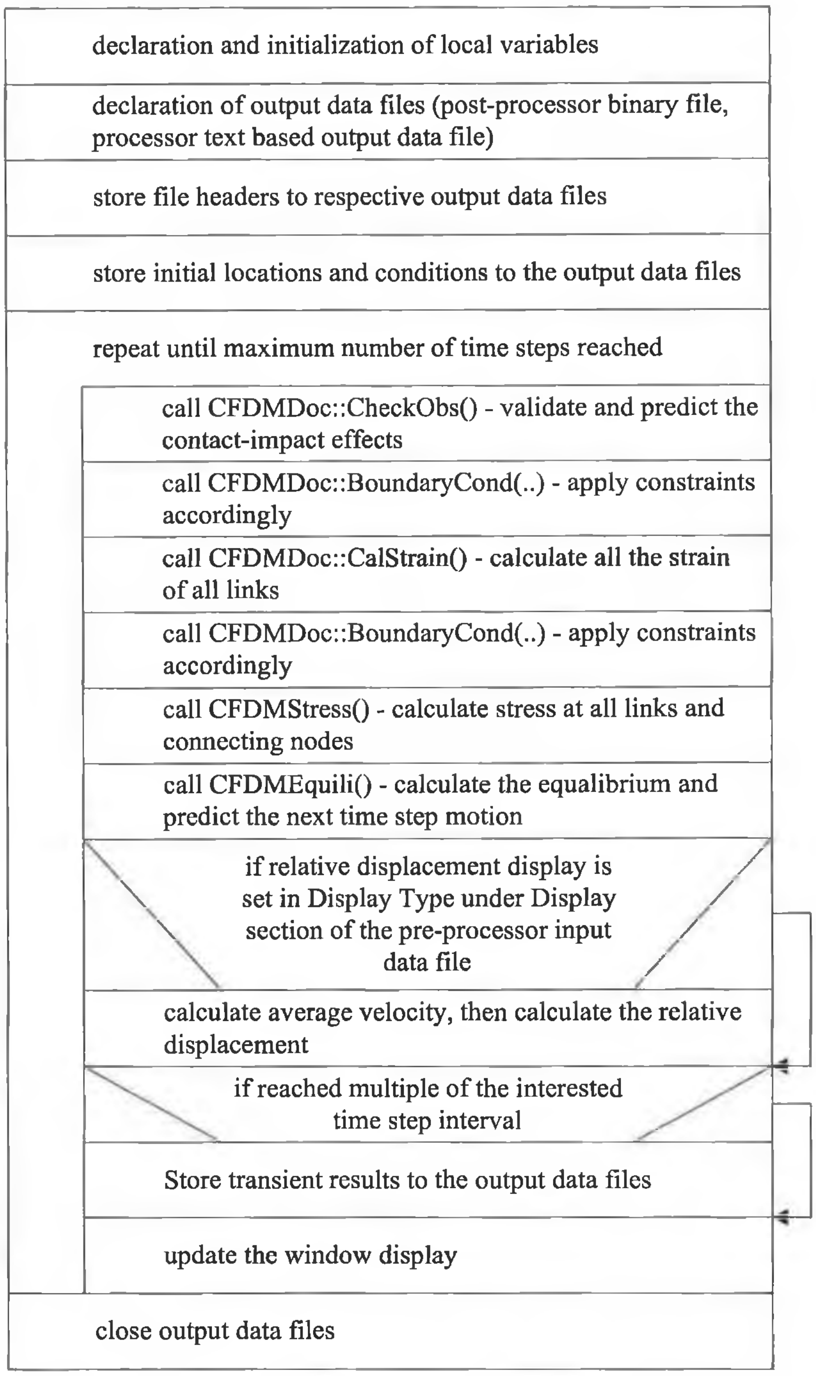


Figure 3-14: ID_FDM command message flow and CFDMVIEW::OnFDM() actions



```
declaration and initialization of local variables
declaration of output data files (post-processor binary file, processor text based output data file)
store file headers to respective output data files
store initial locations and conditions to the output data files
repeat until maximum number of time steps reached
    call CFDMDoc::CheckObs() - validate and predict the contact-impact effects
    call CFDMDoc::BoundaryCond(..) - apply constraints accordingly
    call CFDMDoc::CalStrain() - calculate all the strain of all links
    call CFDMDoc::BoundaryCond(..) - apply constraints accordingly
    call CFDMStress() - calculate stress at all links and connecting nodes
    call CFDMEquili() - calculate the equalibrium and predict the next time step motion
    if relative displacement display is set in Display Type under Display section of the pre-processor input data file
        calculate average velocity, then calculate the relative displacement
    if reached multiple of the interested time step interval
        Store transient results to the output data files
    update the window display
close output data files
```

Figure 3-15: CFDMVIEW::OnFDM() Nassi-Schneiderman chart

## 3.10.2 TEXT-BASED OUTPUT DATA FILE SYSTEM

The file system consists of a file header, section headers, and data contents. The file header provides information about the contents and format of the file system, which is like the following:

**node, x, y, N, Q, M**

No special function is adhered with the file header. All the information of nodes and links are grouped under time instance. Section header is placed at the beginning of every grouped data. The section header tells the corresponding time steps and exact time. The first column is the node/link number, then follows by the location of the x and y co-ordinated of the node, the axial force of the link, and the shear and bending moment at the node position. A typical data file can be found in Appendix A.

## 3.10.3 DISCUSSIONS AND SUMMARIES

The FDM is having typical processor functions. Anyhow, the user can view the transient response of predicted by the processor during the processing. Figure 3-15 shows that the screen will be updated with the latest information (transient response) after storing information to the output data files.

The main drawback of the FDM processor is that, the user does not have any clue when the program will terminate. Furthermore, no way to stop the processing task except to issue a force termination of the application itself. The processor is primitive which only provide the most basic function of processing the data. The development of FDM processor is not based on multithreads method, which leads the FDM eating major part of the computer resources by force. Thus, the performance of other application besides the FDM will dramatically degrade, and the user has no control on it.

The Witmer *et al.* FD model suffers some drawbacks in implementation and modelling limitation. The model will be explicated and the shortcomings will be discussed in the theory section.

## 3.11 FDM POST-PROCESSOR

The FDM post-processor will further interpret the processed data and presents to the user on the screen and/or printer. The present FDM post-processor is able to animate the transient response and also showing some other information such as the bending moment on respective links in contour form.

All the pre-processor functions are embedded in two main classes. They are COptDoc class, a document class inheritance, and COptView class, a view class inheritance. Where generally the document class will handle the input data file (the binary output data file by the FDM processor). The view class will handle the device context and display functions, which is the most important for post-processing. A supportive class, namely COptPathDlg, a dialog box class, is included. The dialog box class is meant to handle a dialog box where a user is asked to select several options in viewing the result of FDM.

### 3.11.1 POST-PROCESSOR MENU SYSTEM

Before illustrating the post-processor features, the menu system has to be revealed. The FDM post-processor menu resource coding is listed under section A.4.2 of Appendix A.

The main functions of FDM post-processor are gathered under the menu *Output Result*. The menu items are as below:

*Displacement Path* – activates the dialog box under COptPathDlg class and display transient response at designated time step.

*Animate Path* – shows animated transient response history, as well as the contour display of selected quantity.

### 3.11.2 COPTDOC CLASS

COptDoc class is publicly inherited from CDocument class. The main and only functions for this class is to retrieve the data from the binary data file created by the FDM processor, and store them to appropriate variables.

The COptDoc stores a great number of variables, which handles up to a maximum of 35 time step intervals. The intervals limit can be altered but have to be realistic. The higher pre-defined limit will cause more computer resources, which is the computer memory storage in this case. The data information retrieved from the binary file consist of overheads and transient data. The size (the contents) of the overheads will not change no matter how many time step intervals are recorded. The overheads can further divide into model-dependent overheads and independent overheads. While the size of transient data will change proportionally to the number of time step intervals recorded.

The maximum and minimum value of x and y co-ordinates, bending moments, shear forces, and axial forces, are stored immediately after retrieving the data. It will be illustrated further in the following section concerning *Serialization* implementation.

To cope with the retrieval process, the COptDoc implement the *Serialization* method similar to the pre-processor (please refer section 3.9.3). *Serialization* here is unidirectional, which only retrieve data and no storing process.

### 3.11.3 COPTVIEW CLASS

COptView class is publicly inherited from CView class. The main function of the COptView is to manipulate the window display of the document, COptDoc. The class consists of six public member variables and four private member variables. They are used to store window size, transient time step number, colour range (the contour) and some Boolean control flags.

The COptView carries a number of member functions and overrides some common operations. All the member functions are performing supportive tasks. The main drawing code is placed in the overridden function, COptView::OnDraw. The device context is customised to automate the screen display orientation and extents, both windows and viewport, in COptView::OnPrepareDC function. Further explication can be found in the implementation of message section.

The COptView class catches five messages. They are the widow size change, a window message, two command messages of handling *ID_OTP_DISPLACEMENT* and *ID_OTP_ANIMATEPATH* (please refer Coding A-4 of Appendix A) and two printing messages. The window message of window size change need to be monitored, as the program needs to scale up or down the co-ordinate system used for display accordingly. Whereas the command messages are used to handle user request of different display options. The last two messages are used to customise printing job.

### 3.11.4 COPTPATHDLG CLASS

The COptPathDlg is a public inherited CDialog, which controls the dialog box with identity of *ID_OTPPATH*. A COptPathDlg instance is created when *ID_OTP_DISPLACEMENT* command message is fired. The command message is captured by COptView class and handled by its member function, OnOtpDisplacement. The instance of COptPathDlg is created then. The whole flow is further illustrated in Section 3.11.6. The COptPathDlg class containing six member variables. They are used for data exchange between the dialog box and the class. The member variables further transferred to COptView member variables.

Figure 3-16 shows the display of *ID_OTPPATH* dialog box. The user can specify the time step interval (Cycle as referred) number, the contour display item (display quantity as referred), vector display item (vector quantity as referred), option to display initial position, and option to display segments connecting the nodes. Vector display has not been implemented in COptView.

### 3.11.5 BINARY DATA FILE SYSTEM

The FDM post-processor supports only one type of data file, namely FDM Processor binary output data file. The data file is generated by the FDM processor and the data is stored in binary form, the simplest form of data representation. The data file is near to illegible with any other applications, including text editor. The details of the file system is documented under section A.3 of Appendix A

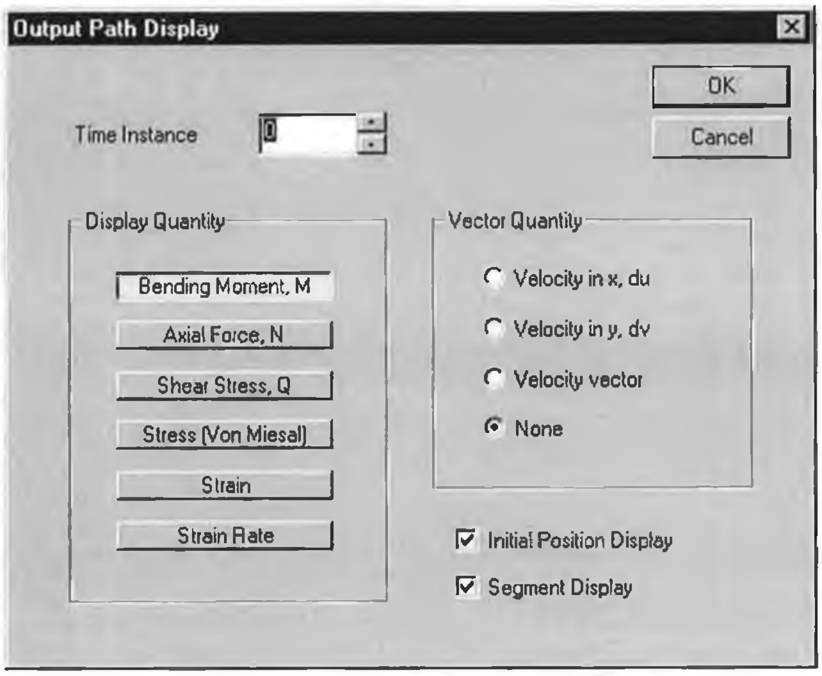


Figure 3-16: ID_OTPPATH dialog box

## 3.11.6 MESSAGE HANDLING

The FDM post-processor messages are handled by COptDoc class and COptView class. Messages are handled separately, where no cross-linking between the two classes during the implementations. Handling of several important messages will be explicated in the following section.

### A. Serialization Message

Similar to the *Serialization* message handled by the pre-processor, the main function is to retrieve data from the data file. For post-processor serialisation, the data file system is as mentioned in section 3.11.5. *Serialize* member function of CObject is overridden by COptDoc::Serialize. The message flow is similar to the one shown in Figure 3-7, except the destination of the data file. The message carries an instance of CArchive class, where the target data file information is stored. The retrieved data will be stored in the

COptDoc member variables. The implementation of the COptDoc::Serialize is shown as the following:

```
retrieve data file header
repeat retrieving point obstacle data until counter = number
of point obstacle number
repeat retrieving segment information until counter =
number of segments
    Retrieve time step and exact time
        retreive nodal displacement and displacement
        change
        retreive link axial force and nodal bending
        moment and shear force
    until counter = number of nodes
until end of file reached
```

Figure 3-17: COptDoc:::Serialize implementation with Nassi-Schneiderman chart

```
set status to "Animation"
initialize counter = 0
while counter <= number of time step intervals
    set display time step interval = counter
    send update view message
    Sleep for 0.4 sec
```

Figure 3-18: COptDoc:::OnOtpAnimatePath implementation with Nassi-Schneiderman chart

### B. Animation Display Command Message

The Animation command message is fired when a user wants to view the transient response of the result through animation. The command message identity representation is *ID_OTP_ANIMATEPATH*. The message is captured by the COptView class, which route to COptView::OnOtpAnimatepath(). The message flow is straightforward and no external data flow. Internal messages are fired from time to time by *COptView::OnOtpAnimatepath* to update the display of the windows. The implementation is simple and shown in the following chart.

### C. Transient Response Display Command Message

A user can view particular transient result. The request is sent by issuing *ID_DISPLACEMENT* command message with selecting *Output Result – Displacement* in the main menu. COptView will catch the message and route it to its member function, *OnOtpDisplacement*. An instance of COtpPathDlg class will be created and the corresponding dialog box, *ID_OTPPATH*, will be called. The user will have to provide the appropriate information such the time step interval and selection the display options. The data from the dialog box will be transferred to COtpPathDlg member variables through *DoDataExchange* function. The status of post-processor is changed to *Displacement*. After that the display of the window will be updated with the selection. Figure 3-19 shows the message flow and the sequence of events through out the process. The numbering in the brackets is showing the event sequence.

### D. Window Display Update Message

Whenever display of the window need to be refreshed, a window message is issued and the COptView class will catch message and route it to the COptView::OnDraw member functions, which overrides the OnDraw function of CView class. The message carries a parameter, a pointer to the Device Context class (CDC), namely *pDC*. The implementation of the OnDraw function can be represented by Nassi-Schneiderman chart shown in Figure 3-20

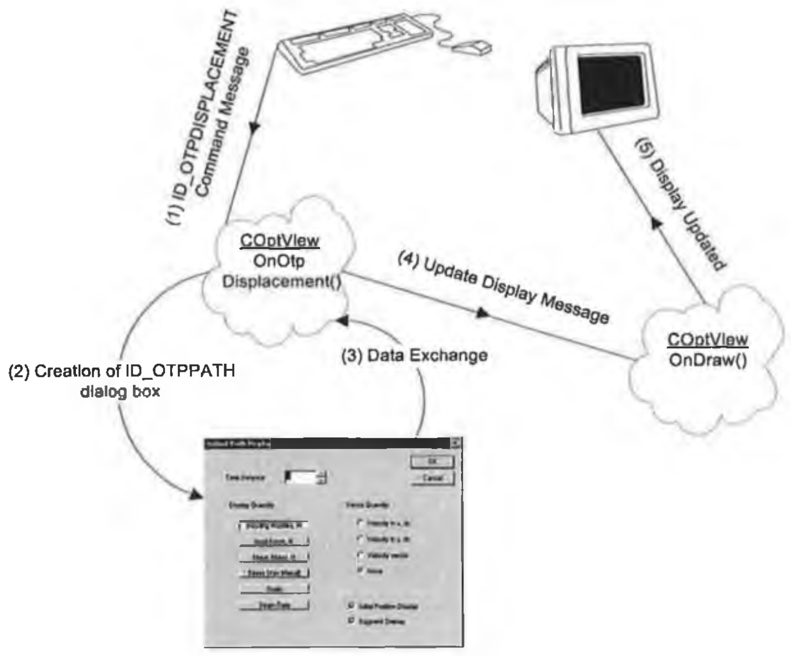

(1) ID_OTPDISPLACEMENT Command Message
COptView OnOtp Displacement()
(2) Creation of ID_OTPPATH dialog box
(3) Data Exchange
(4) Update Display Message
COptView OnDraw()
(5) Display Updated

Figure 3-19: Transient response display command message flow

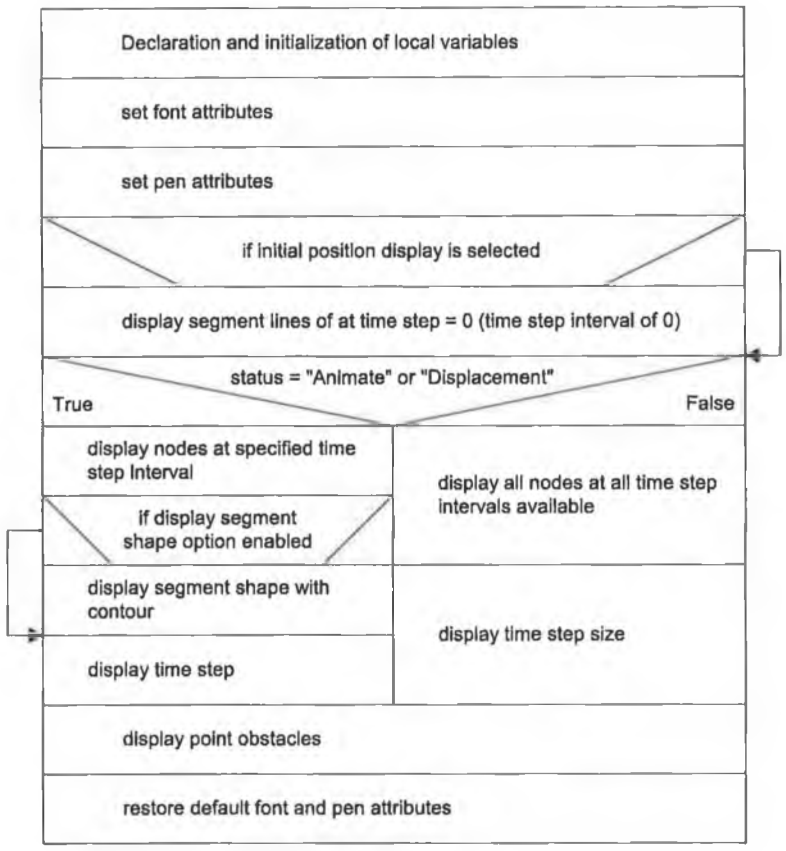

Declaration and initialization of local variables
set font attributes
set pen attributes
if initial position display is selected
display segment lines of at time step = 0 (time step interval of 0)
status = "Animate" or "Displacement"
True
False
display nodes at specified time step interval
if display segment shape option enabled
display segment shape with contour
display time step
display all nodes at all time step intervals available
display time step size
display point obstacles
restore default font and pen attributes

Figure 3-20: Nassi-Schneiderman chart of COptView::OnDraw implementation

While displaying the segment lines, a contour colouring scheme is used to show the user selected property, such as the bending moment. Under the false statement at the above chart, all nodal location will be displayed when a data file is initially loaded.

### 3.11.7 POST-PROCESSOR DISPLAYS

Coding A-1 has been used as an example during explication of FDM pre-processor and FDM processor in previous sections. The yielded results from the FDM processor is loaded into the FDM post-processor, and shown in Figure 3-21. The nodal motion profile is shown, where all locations of nodes are displayed. The initial position of the structure is shown as well, by default. After selecting *Output Result – Displacement* in the main menu, Figure 3-22 is shown. Display of 1st time step interval is selected in the dialog box of *ID_OTPPATH* (refer section 3.11.4). The initial position display option and segment display option is enabled. The contour colour is displaying the respective bending moment. Figure 3-23 shows the transient response at t=0.1 s, for 25k steps.

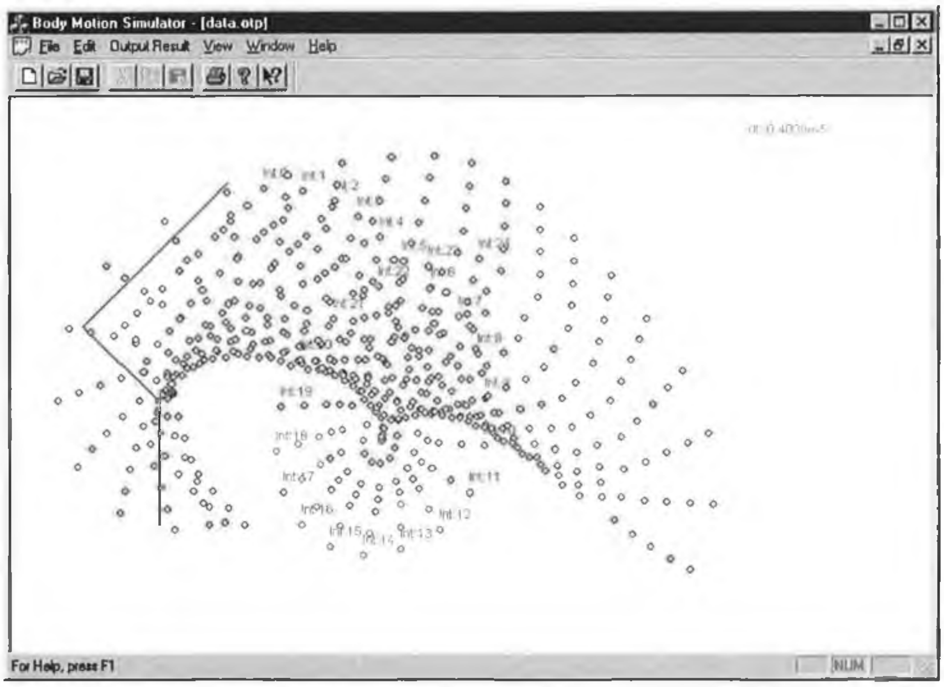


Figure 3-21: Typical initial display after loading the post-processor binary data file

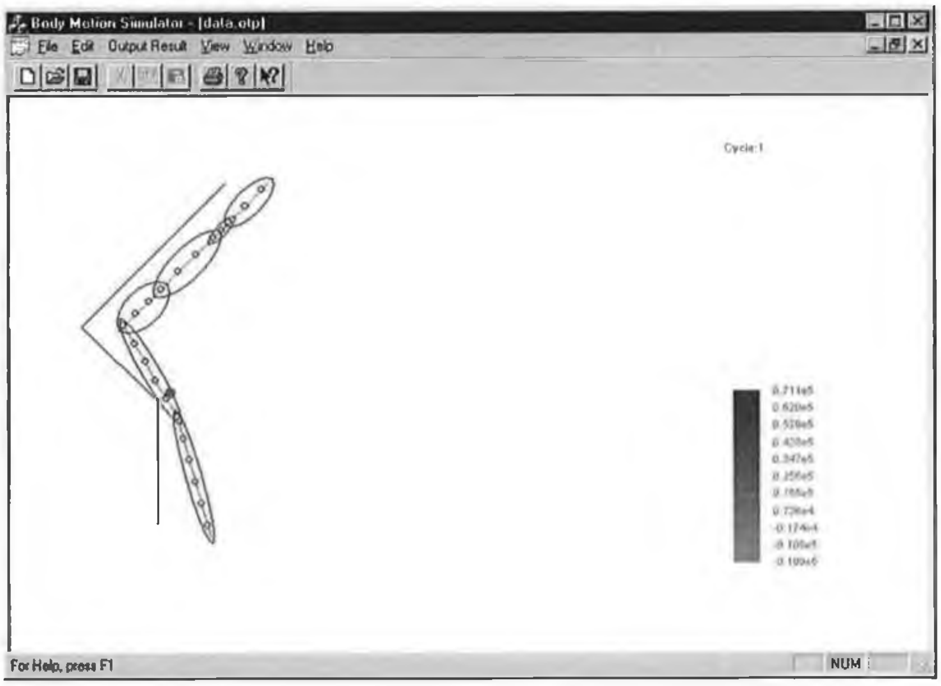


Figure 3-22: Typical frozen transient display of FDM post-processor

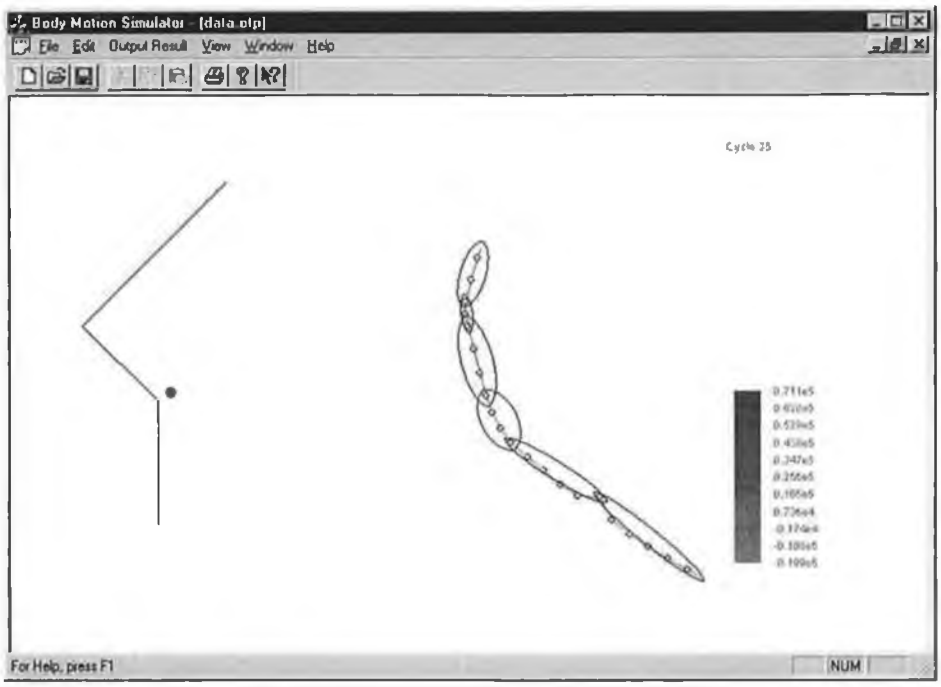


Figure 3-23: Transient response at t=0.1 s of the FDM example.

#### 3.11.8 DISCUSSIONS AND SUMMARIES

The FDM post-processor provides crucial functions, such as animated display of the transient response and contour indication of certain properties. The text information provided is not enough, only the time step interval number is displayed. The printing algorithm is not properly handled as the other drawback of the post-processor. The development of the post-processor is halted and no further improvement has been done, as the whole FDM system was discarded.

## 3.12 FDM APPLICATION METHODOLOGY

To rule out a general methodology in using FDM, a flow chart is presented in Figure 3-24. Before the processor, the methodology has already been shown in previous section. After obtaining results from the processor, a user has to validate the result from both numerical and experimental. Validation process can be done with the FDM post-processor and along with other application software by reading the binary data file and text-based data file respectively.

## 3.13 NUMERICAL EXAMPLE

As an illustration of the FDM, an inverted U block is attached to a solid block travelling at a constant speed of 40 m/s. A sudden ultimate stop at the base block. Figure 3-25 describes the problem and shows the initial conditions. Each segment of the structure is discretised by 5 complete links. The cross-section dimension of the structure is 0.05 m x 0.05 m. The density of the structure is 2710 kg/m$^3$. The material model is elastic – linear elastic-plastic infinite strain, and shown in Figure 3-26. The time step size is set to 2 μs and the maximum number of time steps is 12000 and each time step interval consists of 400 time steps. Coding 3-1 shows the FDM pre-processor input data file. The data file is loaded into the FDM and Figure 3-27 shows the display after *Node & Initial Cond.* has been executed. After being processed by the processor, Figure 3-28 shows the motion profile predicted. The transient responses are shown in Figure 3-29.

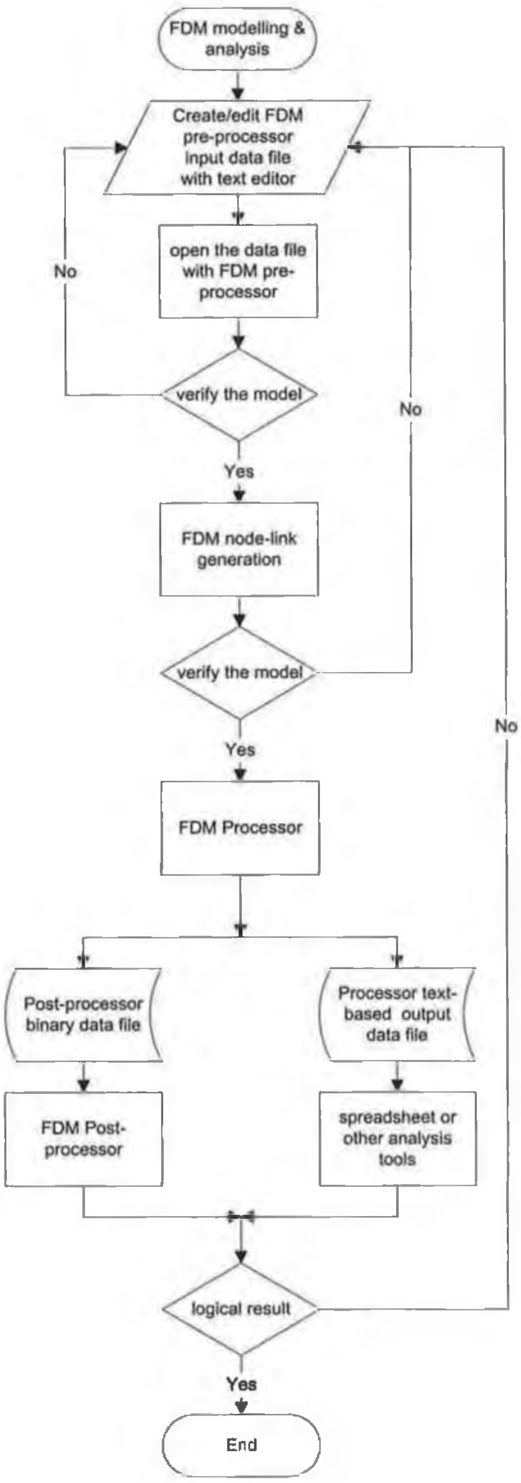

FDM modelling & analysis
Create/edit FDM pre-processor input data file with text editor
open the data file with FDM pre-processor
No
verify the model
Yes
FDM node-link generation
No
verify the model
Yes
No
FDM Processor
Post-processor binary data file
Processor text-based output data file
FDM Post-processor
spreadsheet or other analysis tools
logical result
Yes
End

Figure 3-24: FDM Application Methodology flow chart

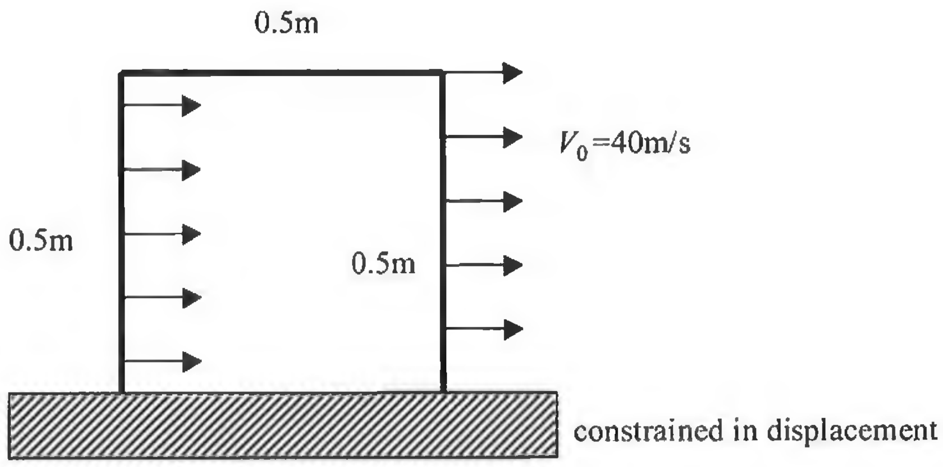


Figure 3-25: Initial conditions of the FDM Numerical example

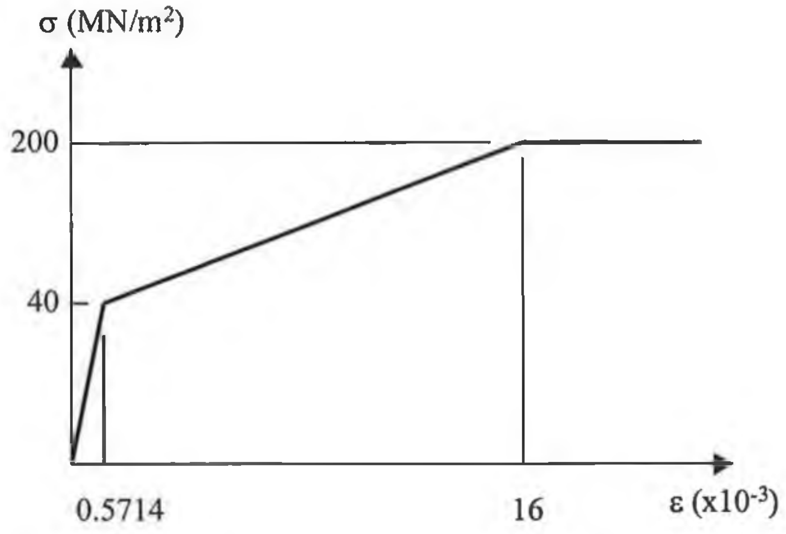


Figure 3-26: Material model for inverted U block problem.

```
wsv

SEGBEG
typ 3;
epc 2,2;
seg 0,     0,    0,     0,     0, 0,   6.0975, 0, 0;
seg seg1, 0,    0.5, 0.05, 0.05, 6, 0.3333, 0, 0;
seg seg2, 0.5, 0.5, 0.05, 0.05, 5, 0.3333, 0, 0;
seg seg3, 0.5, 0,    0.05, 0.05, 5, 0.2333, 0, 0;
SEGEND

OBSBEG
OBSEND

DISBEG
box -0.25, -0.10, 1, 1;
typ 1;
DISEND

EXEBEG
dtm 0.000002;
ncy 12000;
inv 40, 0;
irc 400;
EXEEND
```

Coding 3-1: FDM pre-processor input data file for inverted U block problem

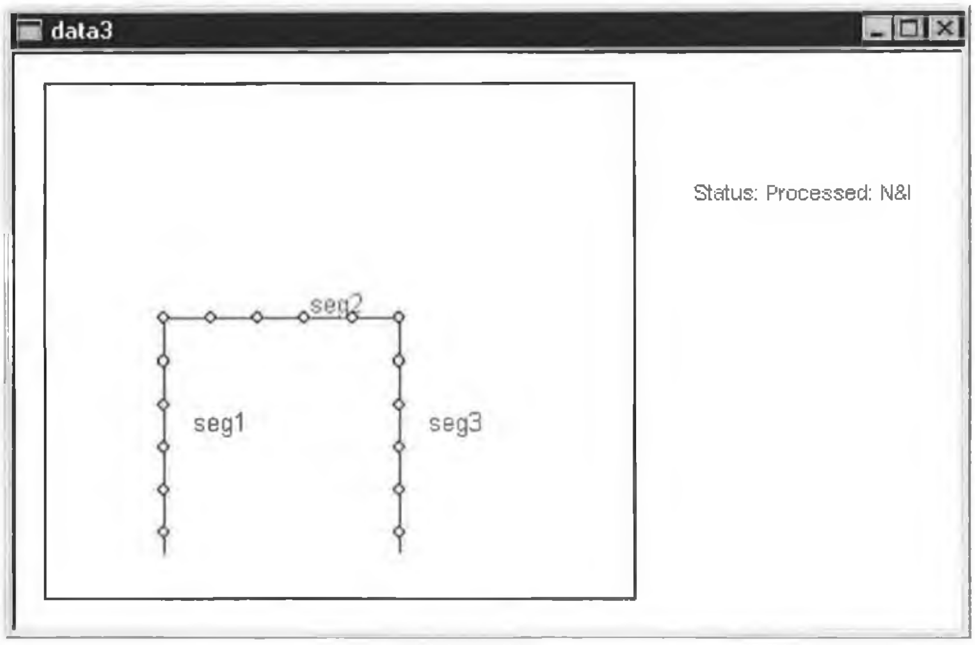


Figure 3-27: Node-link arrangement for inverted U block problem

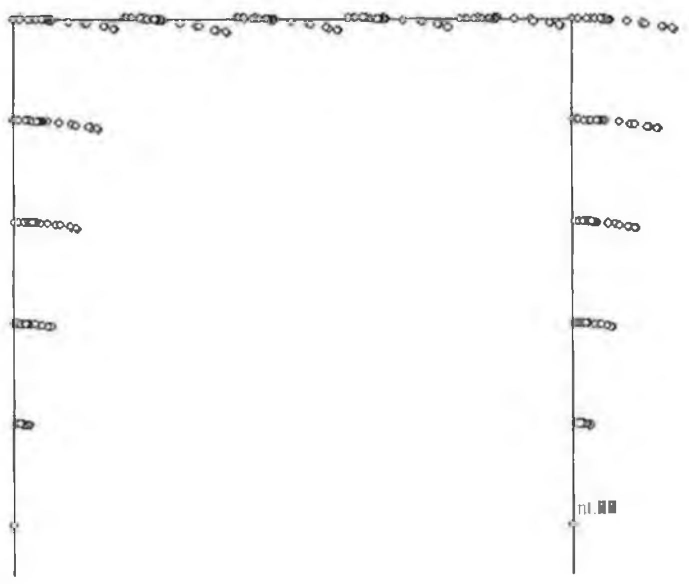


Figure 3-28: Motion profile of inverted U block problem

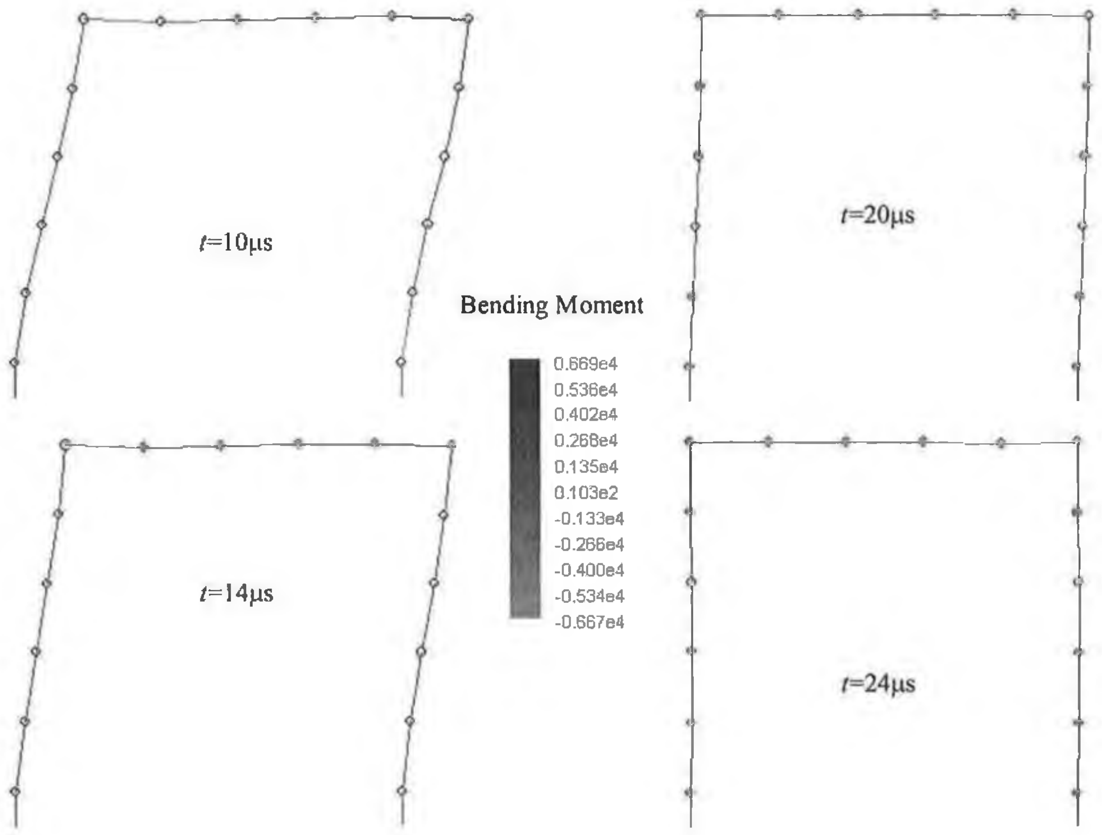


Figure 3-29: Transient response of inverted U block problem

# Chapter 4: Development of Motion Simulation System

## 4.1 Introduction

MSS (Motion Simulation System) is the second generation of numerical-based simulation system developed in this work which is presented this chapter. A general overview of the MSS is followed by component overviews. The development of each component of the MSS is discussed separately. Each component system and data flow, classes development, input-output file systems, and function implementations are presented. A numerical problem is used as an example to illustrate the novelty of the MSS, from pre-processing, processing to post-processing.

## 4.2 MSS Overview

The development of the FDM was constrained due to physical limitations of the FD model (which will be explained further in the theory chapter) used and increasing complexity of the FDM coding. The development of FDM took place in an incremental manner as a large number of basic functions, which were not considered at the planning stage, were added during the course of the development. The lack of knowledge and experience in numerical-based simulation was the main cause. Furthermore, the "introductory" FD code available was a problem-oriented hydrocode, with no pre-post activity. Additions of the pre-post processors were required to cope with the new hydrocode. The "complex" pre-post processors made the amending task more difficult. Thus the development of an entirely new system was decided.

The second generation of the simulation system is named as Motion Simulation System (MSS). The development of the new system is totally different from the FDM. Object-oriented programming techniques have been used throughout the development. The system consists of three distinct components, which were coded separately. Thus, the pre-, post- and the main processor were developed as three independent applications. A user can work on the pre-processing or post-processing while waiting the processor to solve the problem.

Many features have been added into the MSS compare to the FDM. The pre-processor is more automated and a user can develop his/her model directly in the MSS pre-processor without the text editor. No text-based model development is required. Constraint application and initial loading are made easy. The MSS processor incorporated a number of newly developed elements, which make the MSS more versatile. The processor incorporated multithreads technology in which the resources of the computer are divided according to priority, and the other application will use their own space in executing respective tasks. The post-processor is more comprehensive and can generate different text-based output files according to the user requirement, which are to be used for further interpretation.

The new system can be treated as a basis to develop a general-purpose numerical method-based simulation system. OOP makes generalisation of the system to embed other elements possible and simple, which is a traditional shortcomings in FD method, and a strong point in FE method. Around 60 classes were used in the FDM pre-processor development.

The present capability of the MSS processor is set to handle up to 200 nodes, 250 links, 100 contact elements, 15 revolute elements and 15 material models. For the pre-processor, it can handle up to 40 lines, 70 points, 10 arches, and 10 circles. The post-processor is capable of displaying up to 200 time frames.

## 4.3 MSS COMPONENTS AND FUNCTIONS OVERVIEW

Similar to the FDM, the MSS consists of three main components. They are MSS pre-processor, MSS processor, and MSS post-processor. These components have been developed as separate applications, whereas the FDM components were embedded in a single application.

The main function of the pre-processor is to help developing the numerical model. The MSS pre-processor is capable of developing the model from sketch without any text feeding like development of the FDM pre-processor input data file. Lines, circles and arcs can be constructed, meshing (node-link generation in 1D model) mechanism is automated. Different boundary conditions, constraints and initial loading can be applied virtually to any nodes. The MSS can export and import text-based model definition data file, which makes modelling more easy and reusability possible.

The model is solved by the processor. The developed model is loaded and processing of the model is carried out. The user can view the present status or stop the process in the middle of the execution. Multithreads technology has made the processing task being run in parallel with other applications while not bothering much about others. To solve more general numerical problem, two general numerical models were successfully developed and incorporated into the processor. Furthermore, several supportive elements, such as contact-impact algorithm and revolute joint elements, which are developed in this work, were incorporated as well.

The post-processor is used to read the processed output data file of the MSS processor. The post-processor shows the result in graphical form, and animates the transient response history. A user can make special request of text-based output data file and can customise the contents of the data file.

## 4.4 MSS COMPONENT HIERARCHY AND SYSTEM FLOW

The MSS component hierarchy is different from the FDM, where all the components are independent of each other and sit at the same level of the hierarchy. A user can execute any component at any time at any stage of the development. No interaction conflict occurs except one cannot retrieve information from the data file while being processed by the processor.

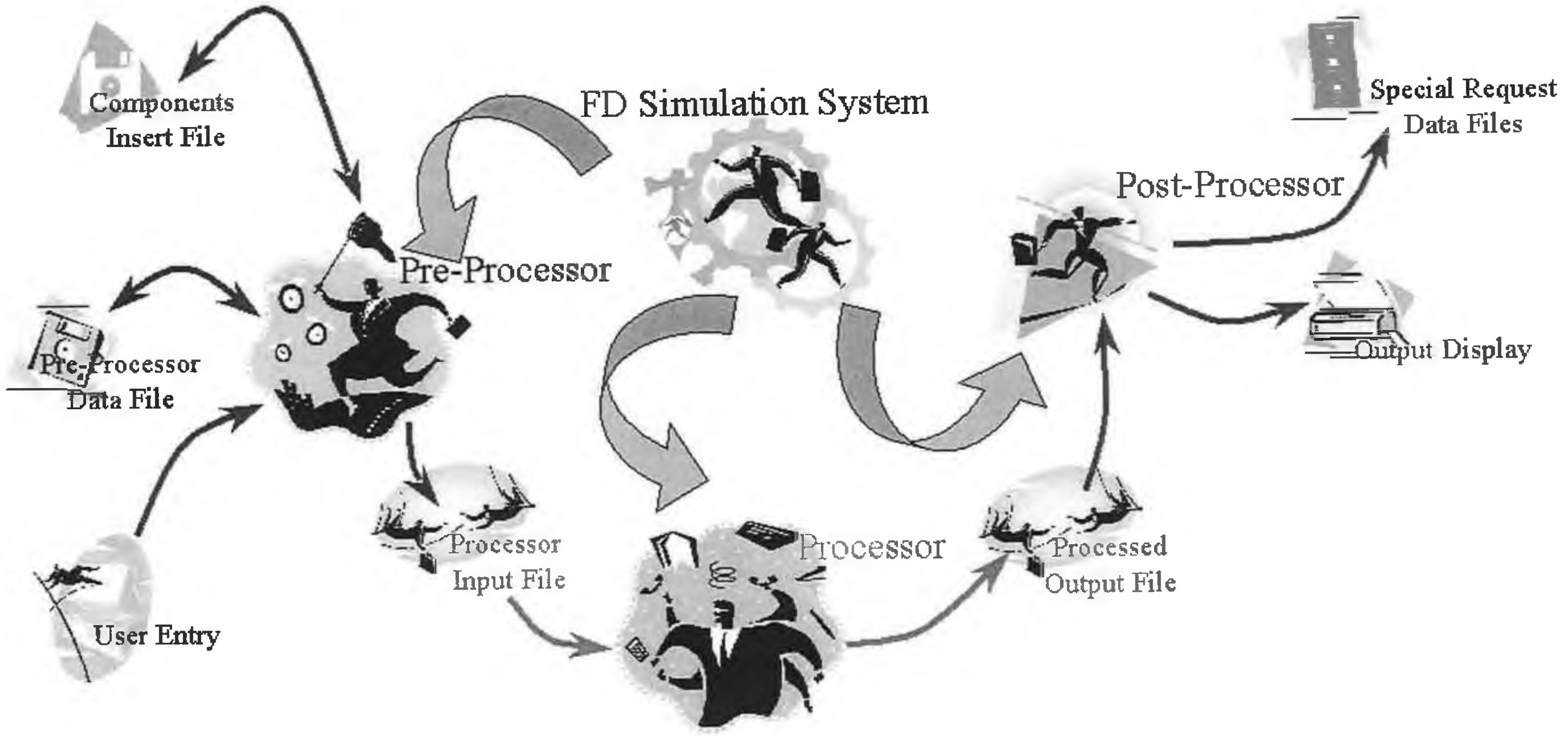


Figure 4-1: MSS system and data flow

Figure 4-1 shows an overview of the MSS system and data flow. The MSS pre-processor accepts two types of data file namely MSS Pre-processor data file and MSS Pre-processor insert file. The first type is in binary format, where all the particulars of the model generated with the pre-processor are stored. The other type is a text-based data file. The user can view and amend the data file, which can be inserted to any existing model, whether it is a blank file, or not. The pre-processor will then generate the node-link relations as well as the nodes and the links themselves. The generated node-link information can be stored in both data files. At the final stage of the pre-processing, an input file for the MSS processor is generated. The MSS processor input file is a text-based data file.

The MSS processor reads the input file and processes it accordingly. An output file is generated, which is called the Processed Output file in binary form. The post-processor reads it and further interprets the result. Special request about particular information can be made and the yield data can be stored in an output file namely special request data file.

## 4.5 MSS PRE-PROCESSOR

The MSS pre-processor is an independent application, named as MSSPre. The main function is to provide a user interface to help developing the numerical model in order to be processed by the MSS processor.

The hierarchy, interaction and data flow among the classes are discussed here briefly. The classes involved will be shown and general discussions will be made. General methodology of developing a model is shown and discussed. Pre-processor handling capabilities are presented and reviewed. Finally, the discussions and summaries will be presented.

### 4.5.1 FUNCTIONS

The MSS pre-processor is a versatile pre-processor compared to the FDM pre-processor. The MSS pre-processor's functions can be divided into three main functional categories and one supportive group. Figure 4-2 shows the MSS pre-processor's overall functions mind map. The pre-processor can use the geometry constructor to build the model. The possible shapes till this stage are point, line, arc and circle. The line constraints are used to help building the model by describing the line with a fixing point, line length and rotary displacement. While constructing lines, arcs, and circles, meshing (node-link generation) information as well as the material information needs to be provided.

Meshing or node-link generation is ready to be executed after completing the geometrical construction. Coinciding nodes may be found after the meshing process. Coinciding nodes have to be eliminated at most of the time. Several mechanisms are available to carry out this task. Material models can be defined at any stage, even before the node-link generation, or even at the initial stage of the pre-processing where the geometrical model has not been constructed.

Boundary conditions may be applied to the nodes and links immediately after the node-link generation or at a later stage. Furthermore, the physical contact constraints can be applied at the same time. Create, edit and delete functions of all the definitions and constructions of both geometrical and node-link must be included. After all the definitions, the model is ready to be processed by the main processor.

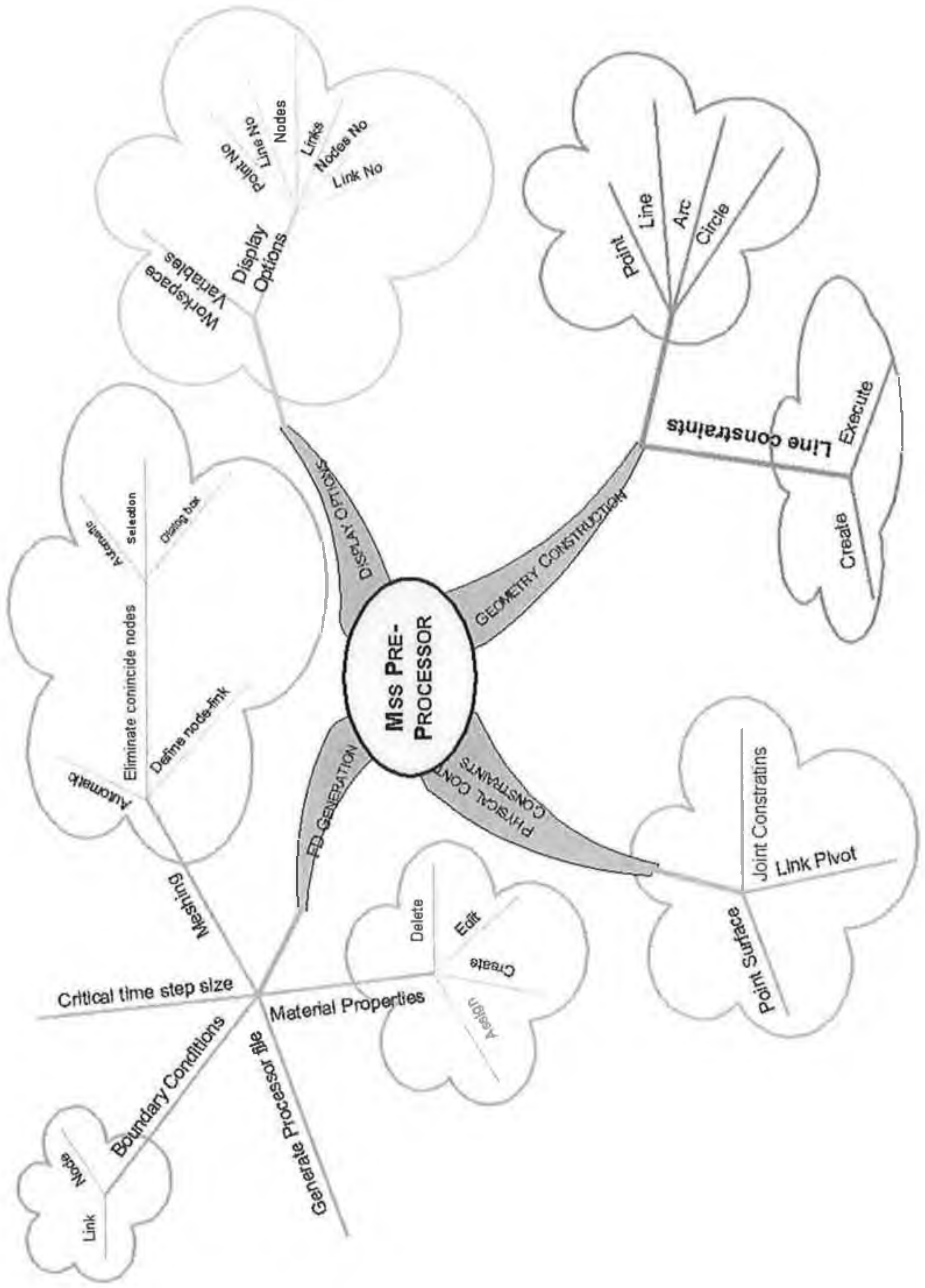


Figure 4-2: MSS pre-processor function map

## 4.5.2 SYSTEM AND DATA FLOW

The system and data flow chart is shown as Figure 4-3. The MSS pre-processor implements Multiple Documents Interface (MDI), where a user can open and access multiple data files at the same time. The data files are stored separately in different documents. Two main classes handle the interfaces of the MSS pre-processor. They are CMSSPreDoc and CMSSPreView classes.

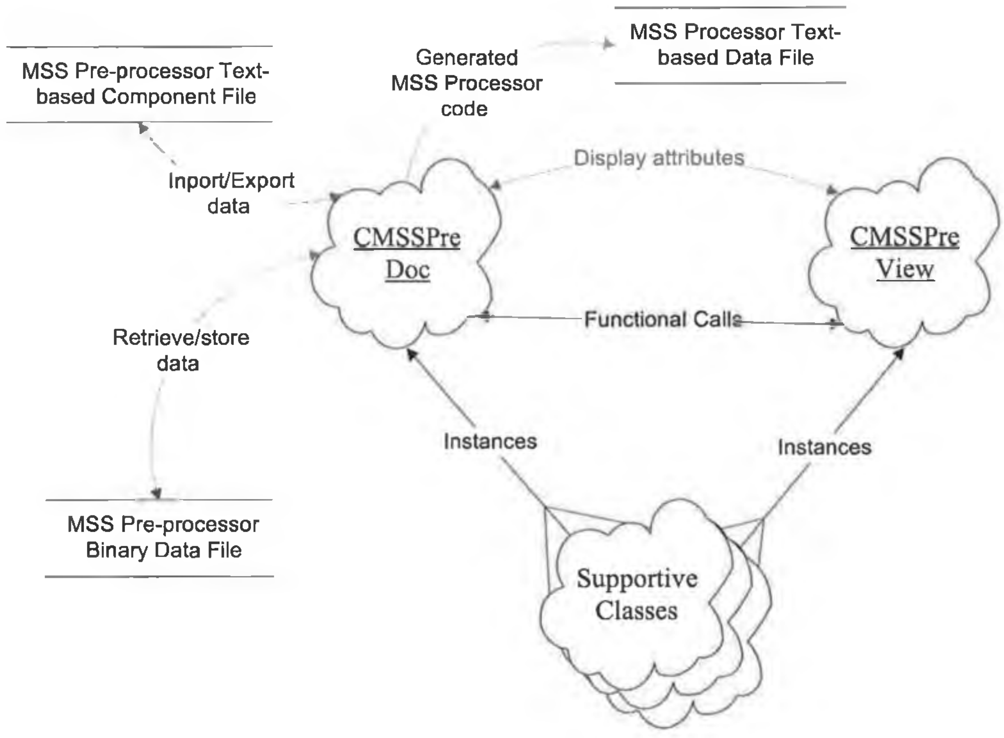

MSS Pre-processor Text-based Component File
Generated MSS Processor code
MSS Processor Text-based Data File
Inport/Export data
Display attributes
CMSSPre Doc
CMSSPre View
Functional Calls
Retrieve/store data
Instances
Instances
MSS Pre-processor Binary Data File
Supportive Classes

Figure 4-3: MSS pre-processor system and data flow

The CMSSPreDoc is a public inheritance of CDocument class, which handles all the data in a data file. The CView class is the ancestor of the CMSSPreView class. The CMSSPreView class is coupled with the CMSSPreDoc at the initialisation of the MSS pre-processor execution. The CMSSPreDoc handles the data input-output between the MSS pre-processor and its supportive file systems. The MSS pre-processor associates three types of data files, namely the MSS Pre-processor Data File, the MSS Pre-processor Component File, and the MSS Processor data file. The MSS Pre-processor extracts and stores data from and to the first two types of data file. The MSS pre-processor only generates the last type of data file.

The CMSSPreView class controls the display functions of the MSS pre-processor. The CMSSPreView uses the data stored in CMSSPreDoc and displays the appropriate information on the screen and as well as printing through the printer. Some member functions of the CMSSPreDoc are called by the CMSSPreView to perform some calculation task. CMSSPreDoc may request a display update through sending message to the CMSSPreView class.

The supportive classes are those helping the implementation of the MSS pre-processor. They can be divided into three main groups, namely Base type, Object type and Interface type. The Base classes serve as a supportive class in the MSS pre-processor, where there is no direct involvement in nature, and can be treated as a new type of variable (data type) and it's implementations. The Object type classes contain the information of the FD model. They may have member variables of Base type classes and/or other Object type classes. Their respective implementation operations are stored inside the class as well. All the classes involving user interface function are gathered under the last group. They can be dealing with dialog boxes, property pages and property sheets. Due to the nature of the pre-processor, the Interface type classes will be of larger number compared to the other two. All the Base and Object classes are designed and developed by the author.

## 4.5.3 CLASSES OVERVIEW AND INTERACTIONS

The classes are divided into 4 main types, which includes the previously mentioned four groups, namely, application, base, interface, and object. A total number of 63 classes is reported and most of them are under interface group, which deal with user interfaces of the pre-processor. All incorporated classes are listed under section B.1 of Appendix B

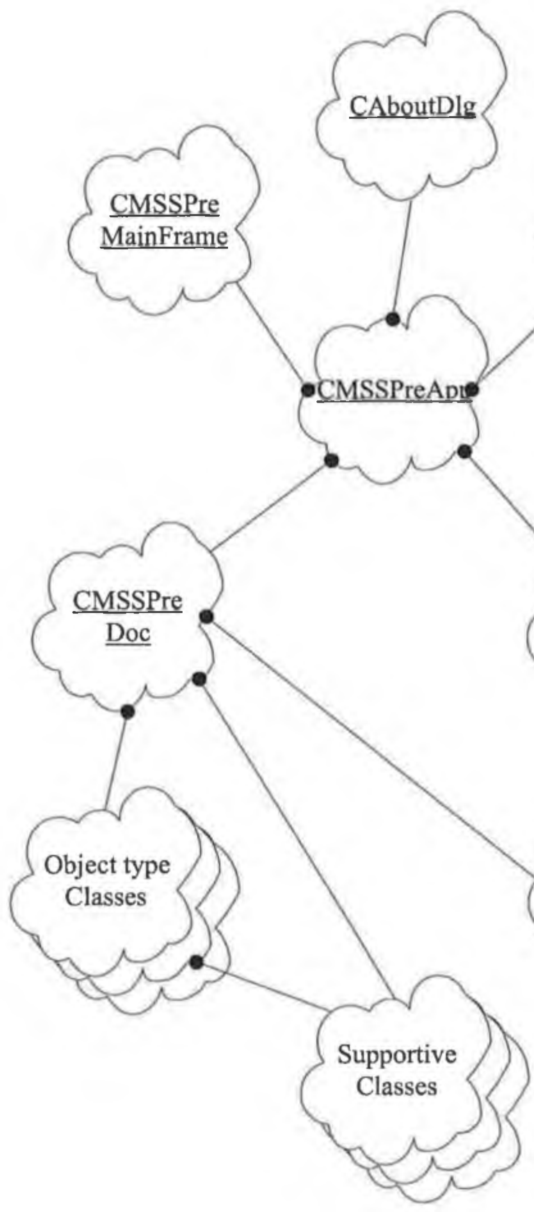


Figure 4-4: MSS pre-processor classes organisation diagram

The class interactions are shown in Figure 4-4. The CMSSPreApp class is the main or the entry class for the whole application. The CMSSPreDoc, CMSSPreView, CMSSPreMaindFrame, and CMSSPreChildFrame are combined together in CMMSApp to handle Multiple Documents Interface (MDI). Instance of CAboutDlg will be created whenever a user requests to display the MSS pre-processor About dialog box. The instance will be destroyed after the closing of the dialog box. Majority of the model information is stored as instances of the Object type classes. The CMSSPreDoc class gathers all the information as its attributes (some may call member variables). As expected, a great number of member variables of these Object type classes are based on the Base type classes. Furthermore, some CMSSPreDoc member variables belong to the Base type classes. For the time being, the CMSSPreView class does not correlate with any Base type and Object type classes. Both CMSSPreDoc and CMSSPreView classes create instances of different Interface type classes, whenever and wherever required.

### 4.5.4 BASE TYPE CLASSES

Based type classes can be treated as low level supporting classes. It gives necessary interpretation of the class type, which can be used by other classes. One can consider these classes as a new type of variables and the instances created are variables with such data type. Only two base type classes are used in the MSS pre-processor. They are CVector and CMatrix classes.

#### A. CVector

The CVector class consists of three attributes, namely *i, j* and *k*. All of them belong to long double data type. The CVector class is used to store vector quantity of its kinds. It may be displacement, velocity, and acceleration of linear or rotary type. The *i, j* and *k* store the quantity in three distinct axes (or direction). It can be in Cartesian co-ordinate system, spherical co-ordinate system or cylindrical co-ordinate system. The value may be represented in global or local co-ordinate system.

#### B. CMatrix

The CMatrix class is used to describe a transformation matrix, which holds 4x4 elements. The matrix elements are long double as data type. The CMatrix is used to assist performing all sort of geometrical transformations associated with the CVector.

### 4.5.5 OBJECT TYPE CLASSES

The Object type classes refer to those classes, which are used to store pre-processing particulars of the model. Different classes are used to describe different entities of the model, such as line, node, contact, etc. A total of 13 classes are grouped under the Object type classes. Only the class attributes are shown in the following sub-sections. The associate operations and implementations will be discussed while dealing the MSS Processor.

The use of various object classes is explained in the following sub-sections. Detail information pertaining object attributes of respective classes can be found in section B.2 of Appendix B.

### 4.5.6 INTERFACE TYPE CLASSES

A total of near to 40 interface type classes are incorporated into the MSS pre-processor. Because of the size and the number, each respective class will not be presented in this thesis. Their use and involvement will be shown during the implementation of the pre-processor at later section.

As a whole, all interface type classes help co-ordinating between the user interface and the main application type classes. Controls of the interface are made within the respective class.

### 4.5.7 MENU DEFINITIONS

Three different menu selections are used in MSS pre-processor. Two of the three are belonging to the window menu bar, where they are used at different stages. The other one is used as a pop-up menu with response to the mouse right button click at particular stage of the execution. Coding of all menus can be found in section B.3 of Appendix B, where menu organisations and respective command message identity (ID) are shown.

## 4.5.8 INPUT-OUTPUT FILE SYSTEMS

As mentioned previously, the MSS pre-processor handles three types of input-output file system, namely MSS Pre-processor Data File, MSS Pre-processor Component file, and MSS Processor Data File. The first two are bi-directional data communication, where both input and output processes are involved. The MSS pre-processor only generates the last type of data file. The first two types will be presented in this section. The last one will be illustrated in the MSS processor Input-Output File System section. Their file extensions are "pre", "prx" and "inp" respectively.

### A. MSS Pre-Processor Data File

The MSS pre-processor data file is in binary form, which mean the data is stored in binary rather than decimal nor text. In order to decode the data file, one needs to know the file systems and interprets accordingly. The purpose of the MSSPre data file is to store the model data created by the MSS pre-processor. The data can be retrieved at a later time, when required. The data file stores everything from working environment setting to model construction, from boundary conditions to initial loading, and from contact elements to revolute joint elements. The MSS pre-processor data file does not store the processor processing parameters, which is stored in the generated MSS Processor Data File. The file system is documented in detail under section B.4 of Appendix B

### B. MSS Pre-Processor Component Data File

One can import or export the entities of whole model through the MSS pre-processor. The purpose of this is to enable combining the present model with some others. The data file is text-based. A user can use the primitive way (like the way used in FDM) to work on the model through a text editor. One can easily amend the contents of the model without doing it in the pre-processor, which may be more tedious.

The MSS pre-processor component data file is pretty much similar to the FDM pre-processor input data file. An improvement has been made, where no strict sequence has to be followed while writing. The user can define *Line* one attributes and followed by *Contact* one, then back to work on *Line* two. Sequence has to be followed while providing the attributes of an entity. A typical Component data file can be found in Appendix B.

File header provides information about the generation of the component data file and is placed at the beginning part of the data file. Entities of same kind are grouped together and brief data representation of respective entity entries is provided as the header of the entity section.

While editing or appending the component data file, the user does not required to follow the sequence of placing the entities, which means, one can place a new *Line* entity at the end of the file, or even under the section header of *Node*. It makes no difference to the MSS pre-processor. But the order of placing attributes of the entities must be correct.

Entity definitions are illustrated under section B.5 of Appendix B.

### 4.5.9 FUNCTIONS IMPLEMENTATION

Individual message handling and implementation will not be presented by it own. This is due to the large amount of messages flying around the MSS pre-processor. Implementation of important functions is explained in details under section B.7 of Appendix B.

### 4.5.10 PRE-PROCESSOR MODELING METHODOLOGY

MSS pre-processor has the capability to perform most of the pre-processing tasks and the modelling methodology is summaried as shown in Figure 4-5. First step of modelling is to develop the geometry model, which consists of points, lines, etc. While creating geometry entities, one can provide the modelling parameters at the same time. Some may prefer to have all the geometry entities readied before going to consider respective modelling parameters. After creation and definition of all geometry entities and respective modelling parameters, one will have to define the material models. In fact, material modelling can be at any stage of the modelling methodology, no straight rules on it. At this stage, the geometry model is ready to undergo node-link generation (may be referred as meshing). After node-link generation, the user should verify the meshed model. If the model does not meshed as required or desired, then check back the modelling parameters provided and re-mesh again. Repeat until satisfied.

After node-link generation, coincide nodes may have to be eliminated. Elimination can be done by using the tools provided by the MSS pre-processor, or do it manually (using a text editor) in the exported text-based component data file, and import back again. Boundary conditions and initial loading can be specified at this stage. Supportive elements such as contact elements and revolute joint elements should then be constructed after this. The model is ready to be processed by the MSS processor now. Before generating the MSS processor data file, one should check the maximum critical time step size allowed.

## 4.5.11 MODELLING EXAMPLE

A numerical problem is used to illustrate the modelling methodology using MSS pre-processor. The problem is a tapered cantilever beam subjected to an impact loading. The problem is shown as Figure 4-6. The cantilever is discretised by 20 nodes. Total impulsive loading of 0.057829 N-s is applied near the tip. The density of the cantilever is 2710 kg/m$^3$. The idealised stress-strain relation is shown in Figure 4-7. Strain rate sensitivity is ignored. Impulsive loading is assumed have equal impact at the nearest two links to the tip of the cantilever, thus the initial velocity of the three nodes of the two links are 34.3895 m/s, 33.5635 m/s and 22.8327m/s (from tip to root).

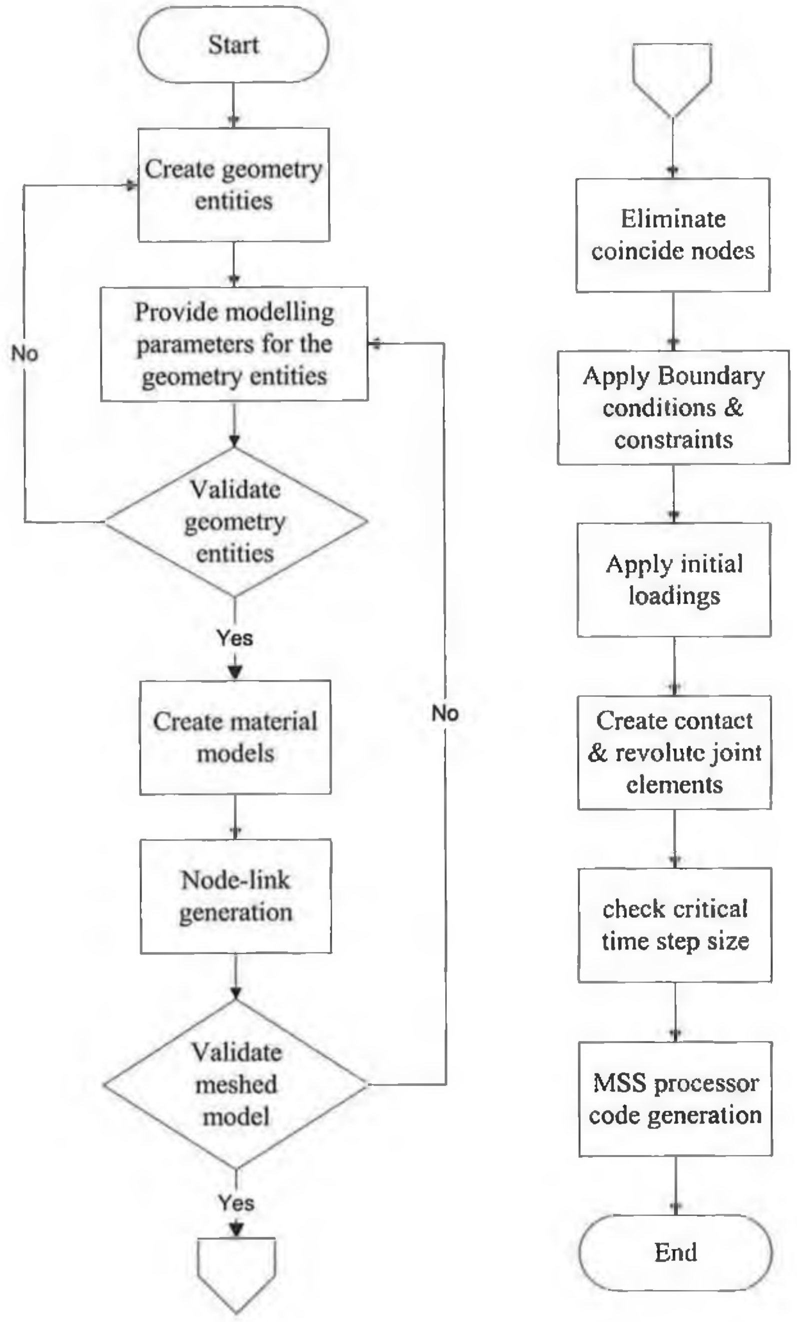

Start
Create geometry entities
Provide modelling parameters for the geometry entities
Validate geometry entities
No
Yes
Create material models
Node-link generation
Validate meshed model
No
Yes
Eliminate coincide nodes
Apply Boundary conditions & constraints
Apply initial loadings
Create contact & revolute joint elements
check critical time step size
MSS processor code generation
End

Figure 4-5: Flow chart of MSS Pre-processor modelling methodology

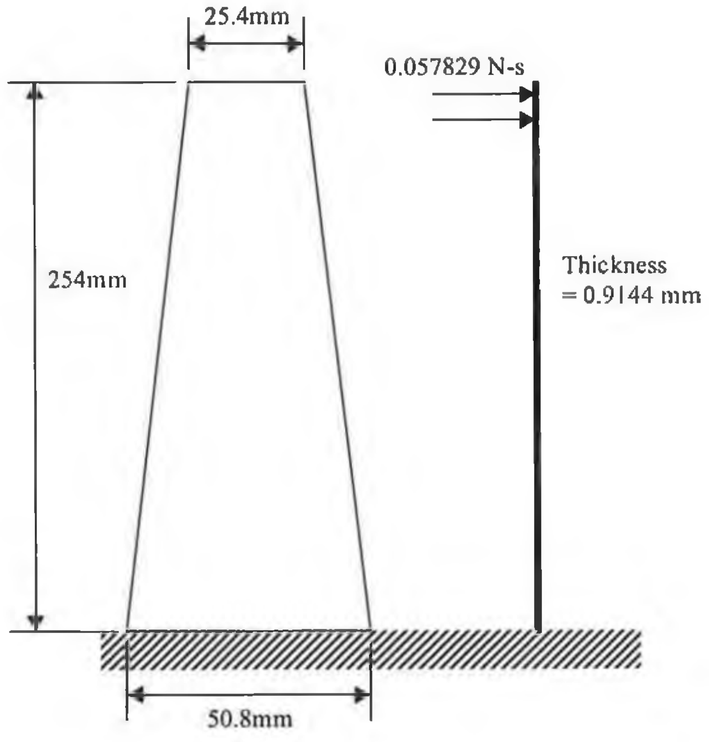


Figure 4-6: Taper cantilever problem

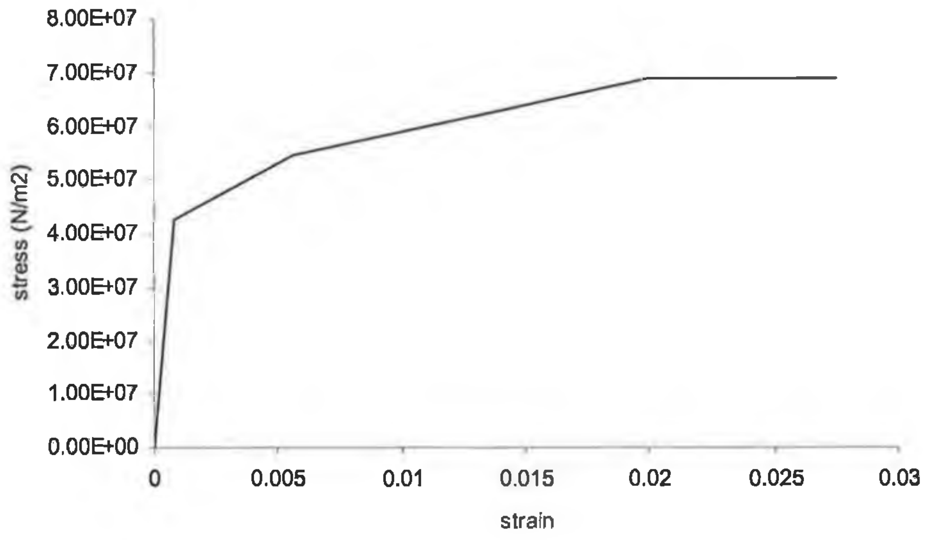


Figure 4-7: Stress-strain relation

The MSS pre-processor is used to model this problem. The first step is to generate geometry entities. Only one geometry entity are required, which is a line. Select line construction icon from the toolbar or *Geometry – Construct – Line* from the main menu. Select *Cord/Cord selected* button, and enter co-ordinates of the line points, which are (0, 0) and (0, 0.254), and press *Ok*. A small vertical line will appeared at the bottom left corner of the window. Change the display settings (Workspace parameter) to scale up the line display. Next step is to change the line modelling parameters. Move the mouse near to the line and click on right mouse button and select *Line* in the pop-up menu. Line editing property pages will be shown. Put 20 in the number of nodes field and 0 as the material number. Select the *Taper* property page and check the taper effect option. Put 0.0508, 0.0254, 0.0009144, 0.0009144 in *width begin, width end, height begin,* and *height end* fields respectively. Material model can be created at this stage. Geometry entity definition is finished and it is readied to generate corresponding nodes and links.

After node-link generation, end link conditions has to be specified. Before this, generated nodes and links can be shown on the display but selecting *nodes* and *links* under *Display* main menu. The root of the cantilever is fixed while the tip is freed. To specify these boundary conditions, move the mouse near to the tip link, and click on right button of the mouse and select *link*. The boundary conditions setting dialog box will be appeared. Select *Fixed* and *1st node*. For the tip link, select *Freed* and *2nd node.*

To specify the initial loading of the three nodes near to the tips, press the nodal initial loading button on the toolbar. Then, select the tip node on the window, and put 34.3895 in the *Vx* field. Perform the same for the 2nd and 3rd nodes and specify 33.5635 and 22.8327 as *Vx* respectively. The MSS pre-processor display will be as shown in Figure 4-8. The model is readied to be processed by MSS processor. Before that, critical time step size should be checked. MSS pre-processor suggests a maximum time step size of 2.7 μs. The exported component data file is shown as Coding 4-1.

```
SV MSSPre exported component data text file
```

```
by MSS Pre-processor version 1.2, copyright reserved, Oct 1999
---------------------------------
Base MSS Pre-processor file:
Generated on Saturday, October 30, 1999  at 16:54:01

*** Points' position ***
- Format: Pt (int PtNo), (double x),(double y);
Pt 0, 0,0;
Pt 1, 0,0.254;

*** Lines' properties ***
- Format: Ln (int LnNo), (int iPt1),(int iPt2),
    (double BBegin),(double BEnd), (double HBegin),(double HEnd),
    (int iNode), (int iMat), (string Desc);
Ln 0, 0,1, 0.0508,0.0254, 0.0009144,0.0009144, 20, 0, ;

*** Nodes defination ***
- Format: Node (int NodeNo), (double x),(double y),(double z)
    (double Vx),(double Vy),(double Vz),
    (double Fx),(double Fy),(double Fz);
Node 0, 0,0,0, 0,0,0, 0,0,0;
Node 1, 0,0.0133684,0, 0,0,0, 0,0,0;
Node 2, 0,0.0267368,0, 0,0,0, 0,0,0;
Node 3, 0,0.0401053,0, 0,0,0, 0,0,0;
Node 4, 0,0.0534737,0, 0,0,0, 0,0,0;
Node 5, 0,0.0668421,0, 0,0,0, 0,0,0;
Node 6, 0,0.0802105,0, 0,0,0, 0,0,0;
Node 7, 0,0.0935789,0, 0,0,0, 0,0,0;
Node 8, 0,0.106947,0, 0,0,0, 0,0,0;
Node 9, 0,0.120316,0, 0,0,0, 0,0,0;
Node 10, 0,0.133684,0, 0,0,0, 0,0,0;
Node 11, 0,0.147053,0, 0,0,0, 0,0,0;
Node 12, 0,0.160421,0, 0,0,0, 0,0,0;
Node 13, 0,0.173789,0, 0,0,0, 0,0,0;
Node 14, 0,0.187158,0, 0,0,0, 0,0,0;
Node 15, 0,0.200526,0, 0,0,0, 0,0,0;
Node 16, 0,0.213895,0, 0,0,0, 0,0,0;
Node 17, 0,0.227263,0, 22.8327,0,0, 0,0,0;
Node 18, 0,0.240632,0, 33.5635,0,0, 0,0,0;
Node 19, 0,0.254,0, 34.3895,0,0, 0,0,0;

*** Links defination ***
- Format: Link (int LkNo), (int iPt1),(int iPt2),
    (double B),(double H), (int iMat);
Link 0, 0,1, 0.0501316,0.0009144, 0;
Link 1, 1,2, 0.0487947,0.0009144, 0;
Link 2, 2,3, 0.0474579,0.0009144, 0;
Link 3, 3,4, 0.0461211,0.0009144, 0;
Link 4, 4,5, 0.0447842,0.0009144, 0;
Link 5, 5,6, 0.0434474,0.0009144, 0;
Link 6, 6,7, 0.0421105,0.0009144, 0;
Link 7, 7,8, 0.0407737,0.0009144, 0;
Link 8, 8,9, 0.0394368,0.0009144, 0;
Link 9, 9,10, 0.0381,0.0009144, 0;
Link 10, 10,11, 0.0367632,0.0009144, 0;
Link 11, 11,12, 0.0354263,0.0009144, 0;
Link 12, 12,13, 0.0340895,0.0009144, 0;
Link 13, 13,14, 0.0327526,0.0009144, 0;
Link 14, 14,15, 0.0314158,0.0009144, 0;
Link 15, 15,16, 0.0300789,0.0009144, 0;
Link 16, 16,17, 0.0287421,0.0009144, 0;
Link 17, 17,18, 0.0274053,0.0009144, 0;
Link 18, 18,19, 0.0260684,0.0009144, 0;

*** End Link conditions ***
```

```
- Format: EndLk (int ELkNo), (int iLk),(int iNode),(int iType);
- iType: 1=fixed, 2=support, 3=free
EndLk 0, 18, 2, 3;
EndLk 1, 0, 1, 1;

*** Material types ***
- Format: Mat (int MatNo), (double Rho),(int iFlg),(int iSubFlg),
   (double D),(double P), (double E1),(double Sigma1), ..;
Mat 0, 2710, 8,3, 1e+006, 0.05, 5.21688e+010,4.25868e+007,
2.45616e+009,5.44165e+007, 1.01157e+009,6.895e+007;

*** END PRX FILE***
```

Coding 4-1: Component data file of tapered cantilever problem

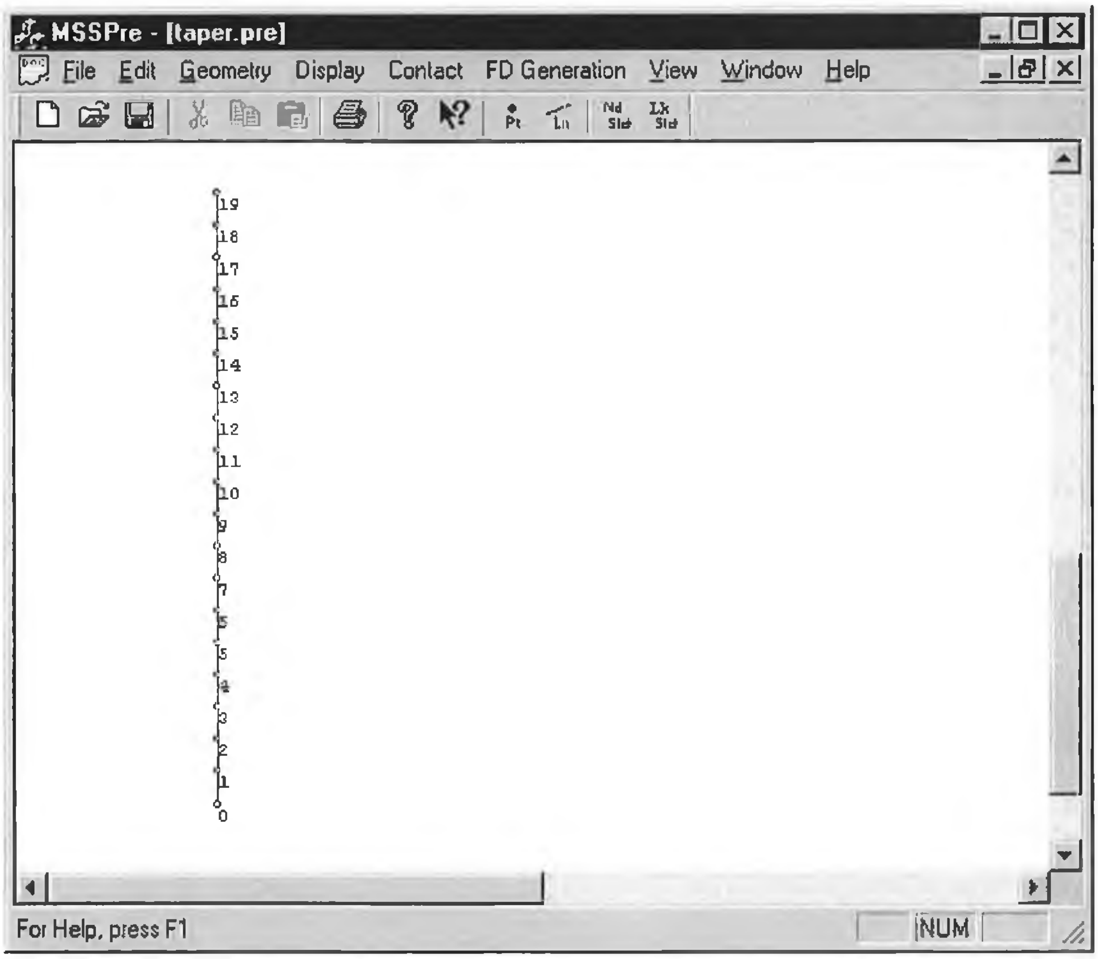


Figure 4-8: MSS pre-processor display of the tapered cantilever problem

### 4.5.12 DISCUSSIONS AND SUMMARIES

MSS pre-processor encompasses a series of construction functions from geometry entity to modelling parameters. All kind of boundary conditions including contact-impact can be defined. The pre-processor is more than enough to deal with one-dimensional element moving in two-dimensional space.

A lot of features are added "on spot" during the development, thus their respective implementation may not be the most optimum nor most efficient. Improvements are still possible towards better design of the classes and handling. For example, the line constraint can further extended to a more powerful utility with more robust constraint validation mechanism. The whole design of the MSS pre-processor left enough space for future expansion to three-dimensional pre-processing. All fundamental elements are based on three-dimensional vectors.

As the whole, the MSS pre-processor is capable of defining:

- Geometry entities, with geometry constraints definition
- Model parameters
- Boundary conditions and initial loadings
- Contact elements
- Revolute joint elements
- Nodes and links construction

The MSS pre-processor is capable of generating nodes and links automatically. Special functions are available to eliminate coincide nodes, which inclusive an automatic method. The MSS pre-processor provides some useful information of selecting the time step size. The critical time step sizes based on various criterions are provided. The pre-processor is capable of exporting and importing model to and from a text-based component data file. One can amend or append the generated data file. Furthermore the exported model can be inserted to any document opened by the MSS pre-processor.

User takes most of the control of the display. Different entities can be selected or hide from the screen. In addition, the display extents (window and viewport extents) can be set manually to suit any special case. Better user-friendliness can be achived by improving the interface of this function.

Majority of the MSS pre-processor functions is new compared to the FDM pre-processor. The functions are mainly to help the user in developing the model, where no text-based model creation is required. While handling of the pre-processor entities, OOP is incorporated, which makes easier future development and leave more space for new entities, elements and features. The use of OOP also promises generalisation. Any new entities, elements and features can be designed and incorporated into the MSS pre-processor with a standard way, and not causing much interruption towards the whole system.

## 4.6 MSS PROCESSOR

The MSS processor is used to process the model, which can be either generated from MSS pre-processor or by manual input data file. The MSS processor accept a new file format defined by the author, which is different compared to both types of input-output data file in the MSS pre-processor as well as any FDM file system.

MSS incorporates different FD models in handling structural impact problems. The FDM model (Witmer *et al.* model) is no longer exist, and replaced by two possible FD models designed and developed by the authors. In additional to that, several new contact elements and new revolute element are incorporated. Anyhow, conventional methods are still available to be used.

The OOP technique is used in the developing process of the MSS processor. This is to maximise the flexibility of the development. Because many unknown supporting functions and features were designed and implemented on the spot, the design of the classes is not at its best form and not in the most efficient way. Furthermore the design and development of the new elements were "incorporated" into the MSS processor simultaneously, thus the MSS has become an experimental platform.

The MSS processor is a multithread application. When the FDM processor is executing, other applications' performance will be effected drastically. While with MSS processor, the computer resources will be diverted to others if they required. But, if no important activity is around, then, 100% of the computer resources will be used. The user reserve the "right" to direct the resources from the MSS processor to any more important process of any on residence applications, or even starting a new application. This features is particular useful if the user only has a computer for his/her work. He/she can let the processor processing the data while he/she can still work on word processing. The user will not feel the existence of the MSS processing tasks behind the screen, and the MSS processor definitely maximises the utilisation of computer resources, especially to process large model and/or time consumed analysis.

### 4.6.1 FUNCTIONS

The MSS processor is used to process the model provided either by the MSS pre-processor or through manual code writing. The responses at different time frame are predicted and the yield information is stored to a processed output data file, which will be further interpreted by the post-processor.

### 4.6.2 SYSTEM AND DATA FLOW

MSS processor is a single document interface (SDI) application. Only one data file (document) is allowed to exist at one instance. To process multiple models at the same time, a user can invoke several MSS processor instances at the same time. The system and data flow chart is shown as Figure 4-9. The MSS processor retrieve data from MSS Processor Input Data File and stored processed data into the MSS Processor Output Data File. The former is a text file, which is normally generated by the MSS pre-processor, but can be created by manual entry with any text editor. The latter is a binary data file, which is generated by the MSS processor. The MSS post-processor will use this output data file for further interpretations. Two main classes handle the interfaces of MSS processor. They are CMSSDoc and CMSSView classes. These two classes play the roles as input document class and view class respectively. The CMSSDoc only retrieves data from the MSS Processor Input Data File. The data will be stored in memory, which will be used by the ThreadProc functions.

The ThreadProc is activated by the MSSDoc class, where a separate thread is used. The main application is executed under its own main thread, while ThreadProc is using the other to perform respective task. The ThreadProc performs MSS Processor processing tasks, while the rest (main application) just provides interfaces between the MSS Processor and the user. The ThreadProc stores processed information to the MSS Processor Output Data File.

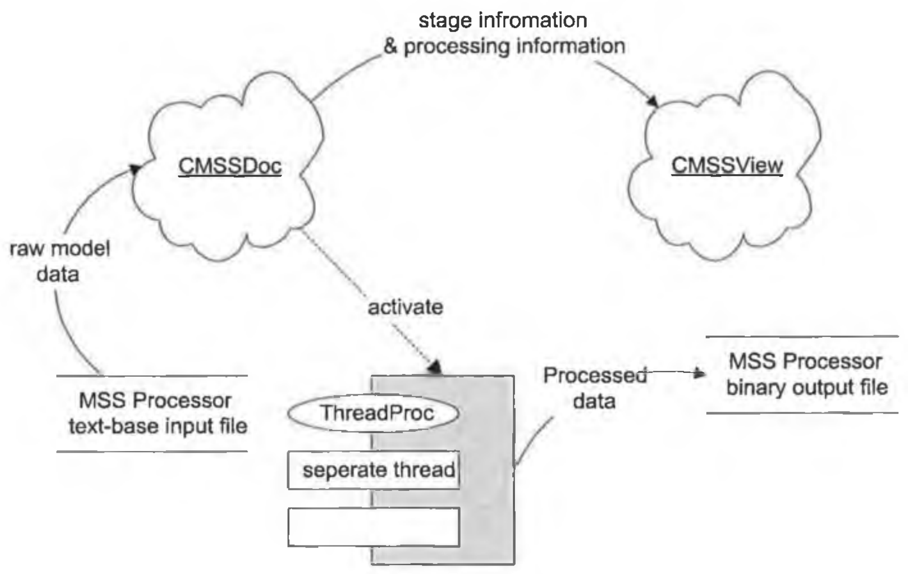


Figure 4-9: MSS system and data flow

The CMSSView controls the display of the window, where the particulars of the MSS processor will be displayed, such as the path of the input data file. While processing, the predicted finishing time will be shown when one updates the display.

## 4.6.3 CLASSES OVERVIEW AND INTERACTIONS

Most of the classes used in the MSS processor store entity attributes and corresponding implementation functions. They are being called and used by the ThreadProc. Similar to the MSS pre-processor documentation, these classes are grouped as *object type* and *base type*. The other will be the standard classes to run an application. The classes interaction is shown in Figure 4-10. The CMSSPreApp class is the main or the entry class for the whole application.

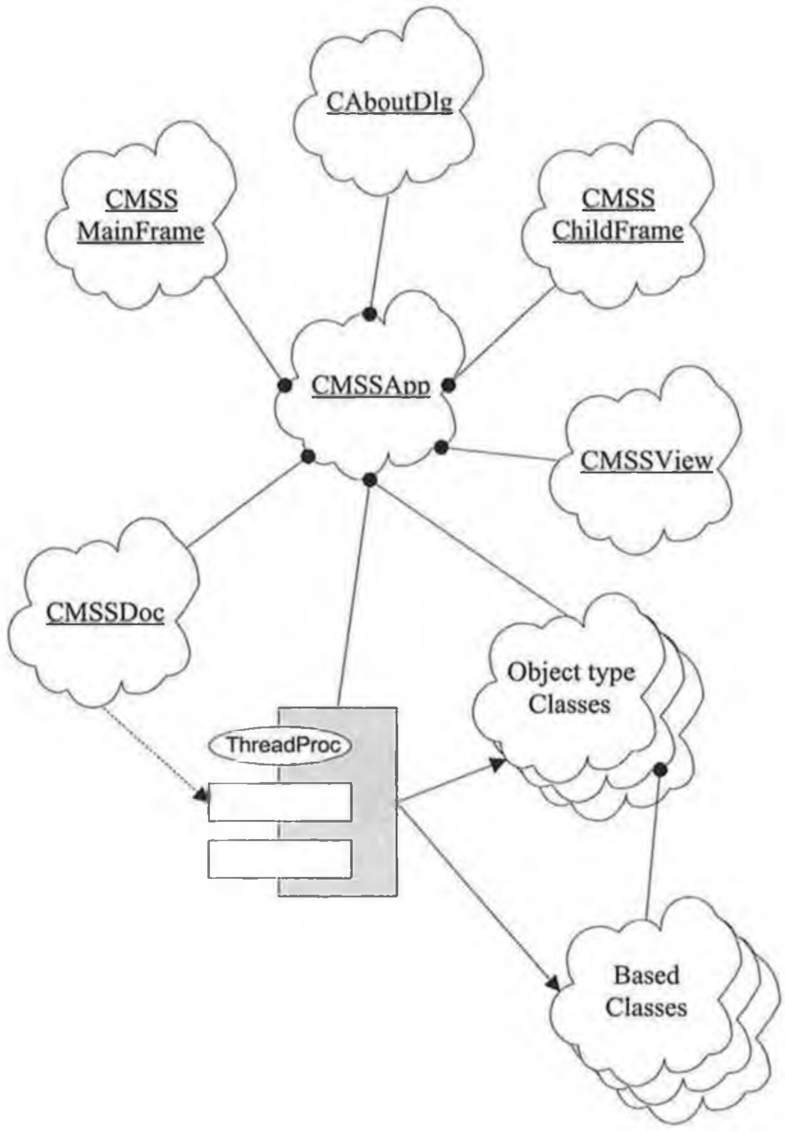


Figure 4-10: MSS Classes organisation

The CMSSPreDoc, CMSSPreView, CMSSPreMaindFrame, and CMSSPreChildFrame are bonded together in CMMSApp to handle document interface. An instance of CAboutDlg will be created whenever a user requests to display the MSS pre-processor About dialog box. Majority of the model information is stored as instances of the Object type classes. The CMSSPreDoc class gather all these information as its attributes (some may call member variables) and stores them as global variables. As expected, a great number of member variables of these Object type classes are based on the Base type classes. A total of 16 classes are involved in the MSS processor development. Descriptions of respective class are listed under section B.8 of Appendix B.

### 4.6.4 BASE TYPE CLASSES

As mentioned previously under MSS pre-processor, Base type classes are used as low level supporting classes. It gives necessary interpretation of the class type, which can be used by other classes. Two main Base type classes are used in the MSS processor. They are CVector and CMatrix classes. They are the same classes as mentioned in section 4.5.4.

#### A. CVector

The CVector class consists of three attributes, namely *i, j* and *k*. All of them are belonging to long double data type. The author has designed a series of functions as well as operators to handle vector implementation.

#### B. CMatrix

The CMatrix class is used to describe transformation matrix, which holds 4x4 elements, long double array with the name *mbr[4][4]*. The CMatrix is used to assist performing all sort of geometrical transformations associated with the CVector class. A number of matrix implementation tools and operators have been developed by this work to cope with its normal functions.

## B.I Matrix – Vector Product

Operator " ^ " is used to represent the product of a matrix and a Vector. The mathematical implementation is as the following:

$$\mathbf{v}' = \mathbf{m}\mathbf{v}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} e_{00} & e_{01} & e_{02} & e_{03} \\ e_{10} & e_{11} & e_{12} & e_{13} \\ e_{20} & e_{21} & e_{22} & e_{23} \\ e_{30} & e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \tag{4-1}$$

The values of $w$ and $w'$ are not important in term of outcome. Their existence is purely for mathematical purpose. Some references simply assuming both values are equal to 1 and so does CMatrix implementation. The only condition is they must not equal to zero. The yielded mathematical result is:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} e_{00}x + e_{01}y + e_{02}z + e_{03}w \\ e_{10}x + e_{11}y + e_{12}z + e_{13}w \\ e_{20}x + e_{21}y + e_{22}z + e_{23}w \\ e_{30}x + e_{31}y + e_{32}z + e_{33}w \end{bmatrix} \tag{4-2}$$

The above is a virtual form used in the CVector and CMatrix implementation. To save memory usage, the $w$ is eradicated from CVector purposely. Thus, Equations (4-1) and (4-2) is changed as the following, which is not valid in mathematical world:

$$\mathbf{v}' = \mathbf{m}\wedge\mathbf{v}$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} e_{00} & e_{01} & e_{02} & e_{03} \\ e_{10} & e_{11} & e_{12} & e_{13} \\ e_{20} & e_{21} & e_{22} & e_{23} \end{bmatrix} \wedge \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4-3}$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} e_{00}x + e_{01}y + e_{02}z + e_{03} \\ e_{10}x + e_{11}y + e_{12}z + e_{13} \\ e_{20}x + e_{21}y + e_{22}z + e_{23} \end{bmatrix}$$

## B.II Rotation Matrix

In three dimensions, there are independent rotations about all three axes, x, y, and z. To define a rotational matrix, one can use the following functions to help.

void **Rx** (long double *Theta*);

void **Ry** (long double *Theta*);

void **Rz** (long double *Theta*);

*where*

**Rx**, **Ry**, **Rz** – member functions of CMatrix class. They define rotation about x-, y- andz-axes.

*Theta* – rotation angle in radian.

*Mathematical Definitions:*

$$\mathbf{R}_x(\theta)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\theta & -\sin\theta & 0\\ 0 & \sin\theta & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{4-4}$$

$$\mathbf{R}_y(\theta)=\begin{bmatrix}\cos\theta & 0 & \sin\theta & 0\\ 0 & 1 & 0 & 0\\ -\sin\theta & 0 & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{4-5}$$

$$\mathbf{R}_z(\theta)=\begin{bmatrix}\cos\theta & -\sin\theta & 0 & 0\\ \sin\theta & \cos\theta & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{4-6}$$

### B.III Translation Matrix

In three dimensions, translation involves displacement in all three member variables of a vector class. The translation matrix definition can be achieved with either of the following syntax:

void **Txyz** (long double *dx*, long double *dy*, long double *dz*);

void **Txyz** (const CVector& *vec*);

*where*

*dx, dy, zy* – translation quantity.

*vec* – a member of CVector, which its member variables, *i, j, and k*, are used to define the translation magnitude and direction.

*Mathematical Definitions:*

$$\mathbf{T}_{\mathrm{xyz}}(\Delta x, \Delta y, \Delta z) = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4-7}$$

## 4.6.5 OBJECT TYPE CLASSES

All object classes described in Table B-6 of Appendix B are very much similar to the corresponding class used in the MSS pre-processor. One can consider Object type classes of the MSS pre-processor are the base classes of respective MSS processor's object type classes. The Object type classes of the MSS processor usually include their associate implementation functions, where do not required with the MSS pre-processor. Thus, the reader is advised to visit Appendix A for details of the class attributes.

Implementation details of member functions of respective classes will not be explicated in this thesis, owing to the great number of member functions involved.

### 4.6.6 MENU DEFINITIONS

Only one menu system exists in the MSS pre-processor. The MSS processor is using the SDI thus no additional main menu system is required. The detail of the menu system is recorded under section B.9 of Appendix B.

The main menu has common items except under *Run*. The pop-up *Run* contains 3 functions, namely *Start, Stop, and Status.* They are used to start, end, and display the status of the processing.

### 4.6.7 INPUT-OUTPUT FILE SYSTEMS

The MSS processor handles an input data file and an output data file types. The input data file is a text file, while the other is a binary data file. The former file system has "inp" as its file extension. "otp" is the extension for the latter file system. The MSS processor reads the data from any input data file and processes it. The MSS processor generates output data file, which stores the predicted transient information. Only the former file system will be described in this section, while the latter will be covered under the MSS post-processor documentation. The MSS processor text-based input data file uses same definition of certain modelling entities as that of the MSS pre-processor component data file. They are *Node, Link, Spoint, EndLk, LkPivot, RMmt, Ctact,* and *Mat.*

There are some additional modelling parameters, which required by the MSS processor but not by the MSS pre-processor. They are defined in the MSS processor input data file with the following syntax:

**Itr** ld *dt*, ld *maxTS*, ld *TSInt*, int *bGrav*;

Where

*dt* – time step size in second.

*maxTS* – maximum number of time step.

*TSInt* – time step interval.

*bGrav*- gravitational acceleration effect option. The option can have the following value:

- 0 no gravitational acceleration effect
- 1 g = -9.81m/s in y-axis

## 4.6.8 FUNCTIONS IMPLEMENTATION

The most important functional implementation is the ThreadProc of MSS. All the processing algorithms are residents within the ThreadProc and/or called by ThreadProc. Other important functions of the MSS are activation and termination of the ThreadProc. Last but not least, the status display and predicted remaining time.

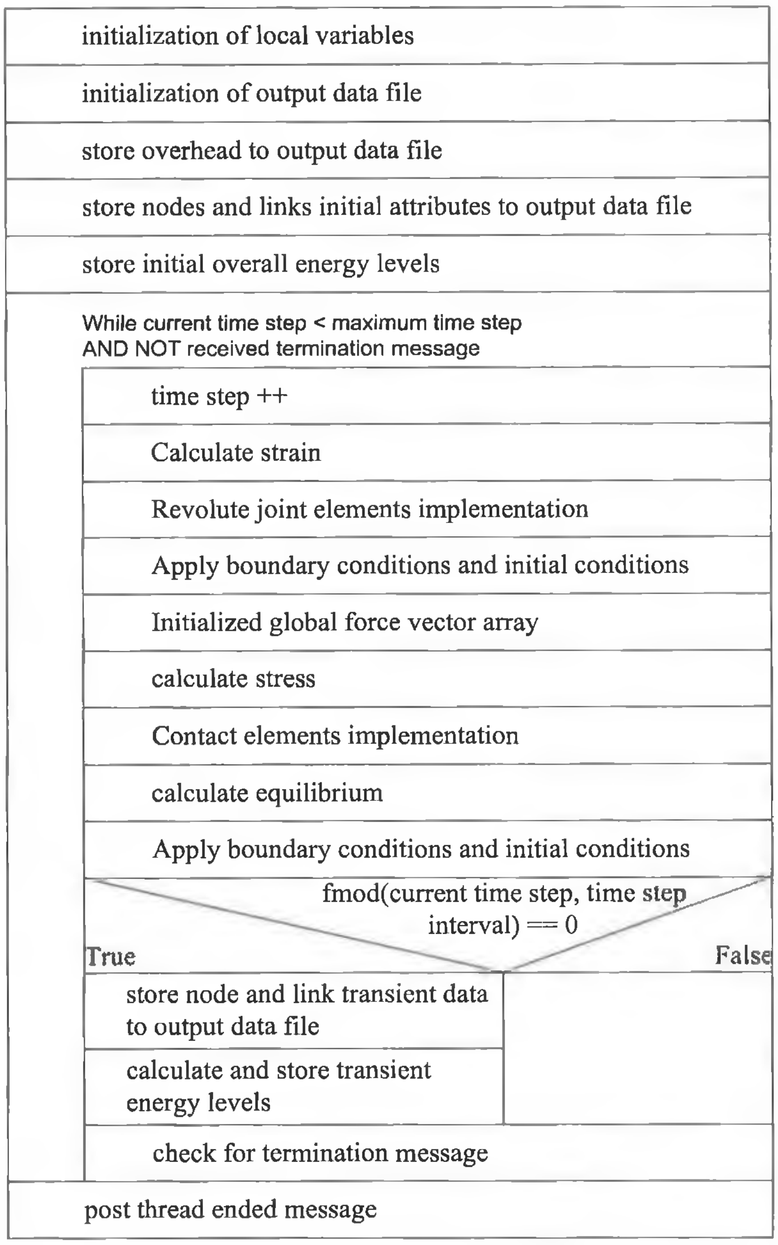
### A. Start Processing

The MSS processing will be activated after the user fires *ID_Start* command message, which is selecting *Run – Start* in the main menu. The menu item is only valid if the MSS processor has read an input data file. Upon initiation, a separate thread is created with thread priority set to the lowest (but not ideal). Then the ThreadProc procedure is assigned to the thread. The *Status* of the MSS processor is set to "Processing" and the display of the windows is updated.

### B. ThreadProc – Processing

MSS processing occupies a lot of computer resources, both microprocessor and memory, all most as many as possible. Thus, it may lead to a vegetative state of the computer while running the processing procedures. This is particularly true while running the application with the lowest level computer, the personal computer. With multitasking capability of Windows 9x and Windows NT, the processing tasks can co-exist with others. Without proper handling of the microprocessor utilisation, the above problem may still occurred on Windows 9x and Windows NT platforms. The author uses multithread method to solve such problem. The algorithms of the MSS processing are shown in Figure 4-11. All of them are under the ThreadProc hierarchy, which belong to a separate thread from the application. The ThreadProc starts with declaration and initialisation of local variables as well as the output data file. All the data file overheads are stored. The overheads are the time step size, number of nodes, number of links, etc. Initial location and conditions of all nodes and links are stored as well. Initial energies level of initial kinetic energy, elastic energy and plastic energy are calculated and stored. The iteration of processing each time step then begins. A counter is used to record the current time step being processed. The iteration continues until the maximum number of time steps reached or receiving termination request from the user. The processing algorithms include calculating of strain, stress and equilibrium. The contact elements and revolute joint elements are handled as well. The next time step displacement will be predicted under the equilibrium calculations. Detail formulation and theory behind of all these will be shown in Chapter 5.

The predicted transient responses are recorded under the MSS data output file at every time step interval. This can be achieved by using *fmod*. The *fmod* function returns the decimal point value of the division of the first parameter over the second. Before ending each iteration, the ThreadProc will try to capture the termination message, a customised message by MSS. Before termination of the thread, a message will be posted to the MSS application where post-processing tasks in MSS can be carried out.



```
initialization of local variables
initialization of output data file
store overhead to output data file
store nodes and links initial attributes to output data file
store initial overall energy levels
While current time step < maximum time step
AND NOT received termination message
    time step ++
    Calculate strain
    Revolute joint elements implementation
    Apply boundary conditions and initial conditions
    Initialized global force vector array
    calculate stress
    Contact elements implementation
    calculate equilibrium
    Apply boundary conditions and initial conditions
    fmod(current time step, time step interval) == 0
    True
        store node and link transient data to output data file
        calculate and store transient energy levels
    False
    check for termination message
post thread ended message
```

Figure 4-11: ThreadProc Nassi-Schneiderman chart

## C. Stop ThreadProc

The command message *ID_STOP* is used to terminate the on-going processing task. The corresponding menu item and the toolbar button are only valid when the ThreadProc is running. When the *ID_STOP* is fired, the CMSSView class will capture it and route to the OnStop member function. The thread termination message will then be fired, and ThreadProc will capture the message.

### D. Show MSS Status

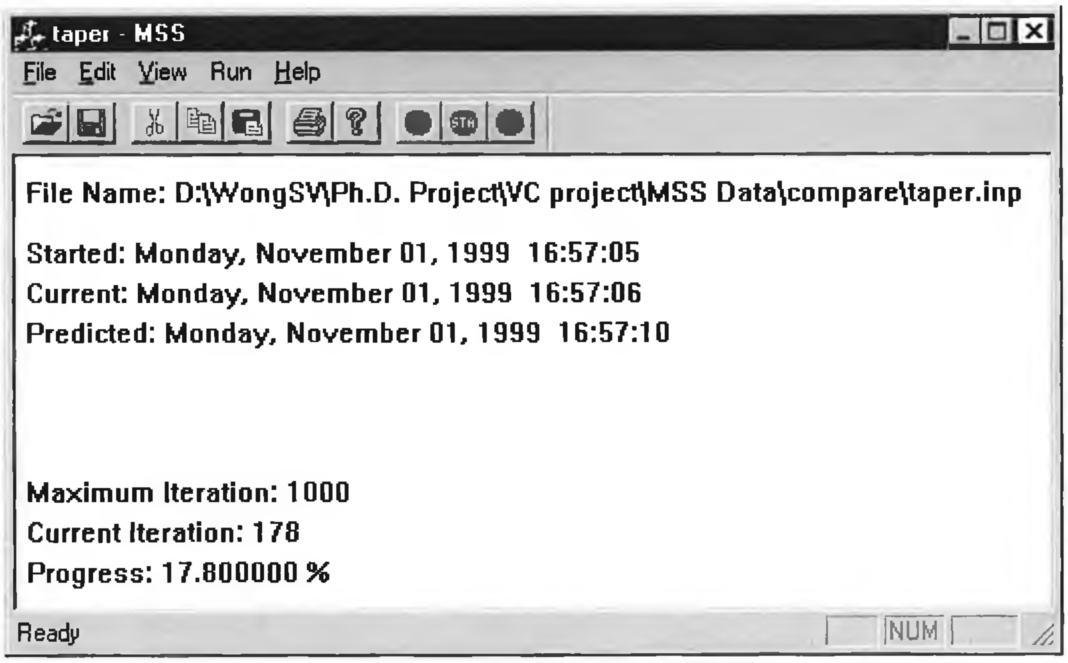
A user can check the present processing status. The predicted finishing time will be calculated and shown on the window. Command message *ID_STATUS* is used to handle the status showing request. The message handling function will predict the finishing processing time and update the display of the window. The finishing processing time is calculated with the following equation:

$$T_{\text{end}} = \frac{T_{\text{elapsed}}}{crTS} maxTS \tag{4-8}$$

where $T_{\text{end}}$ is predicted finishing time. $T_{\text{elapsed}}$, $crTS$, and $maxTS$ are the time elapsed, current time step and the maximum number of time steps respectively.

## 4.6.9 MSS PROCESSOR DISPLAYS PREVIEW

The MSS pre-processor example, tapered cantilever beam problem, is processed in the MSS processor. The generated MSS Processor Input Data File is read and *Start* button is pressed. Figure 4-12 shows the display of the MSS processing the data file after *Status* button is pressed. The next figure shows the display of the window after the MSS finished solving the problem.

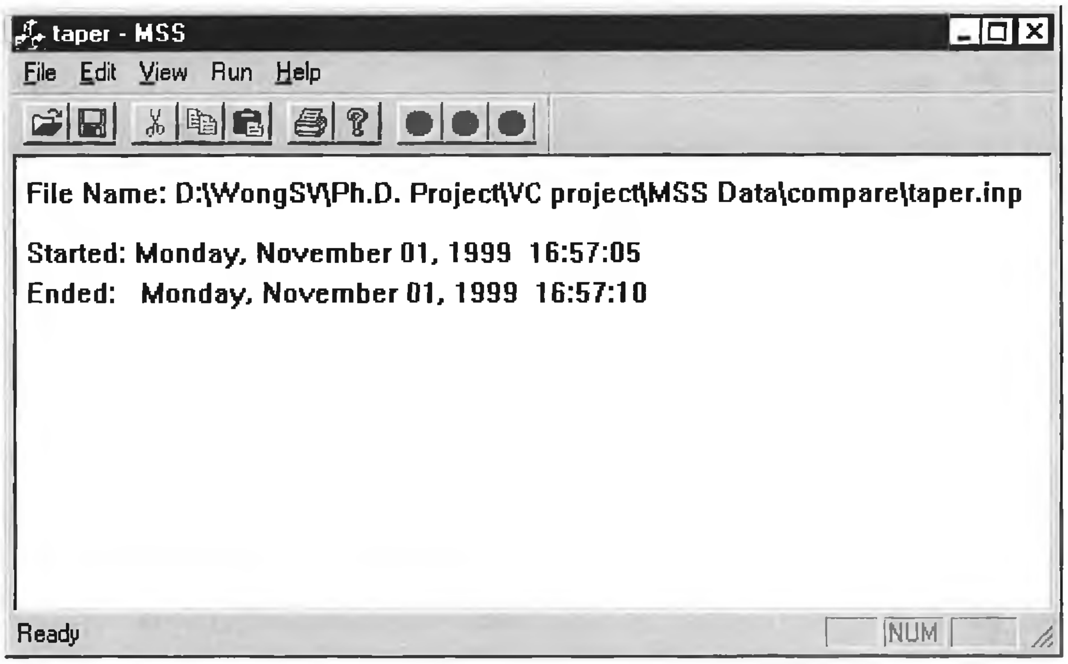


File Name: D:\WongSV\Ph.D. Project\VC project\MSS Data\compare\taper.inp

Started: Monday, November 01, 1999 16:57:05
Current: Monday, November 01, 1999 16:57:06
Predicted: Monday, November 01, 1999 16:57:10

Maximum Iteration: 1000
Current Iteration: 178
Progress: 17.800000 %

Figure 4-12: MSS window display while processing the input data file



File Name: D:\WongSV\Ph.D. Project\VC project\MSS Data\compare\taper.inp

Started: Monday, November 01, 1999 16:57:05
Ended: Monday, November 01, 1999 16:57:10

Figure 4-13: MSS window display after processed the input data file

### 4.6.10 DISCUSSIONS AND SUMMARIES

The MSS processor is more flexible compared to the FDM processor. Several new programming methods and technologies are incorporated. Among them are using multithreads, and object-oriented programming. The MSS is using different FD models, different contact elements, and revolute joint element, which are designed and developed by the author. They are more flexible and more suitable to be used as general elements.

The MSS processor creates separate thread to process the problem. This has made the MSS application more flexible and efficient. A user can let the processor to work on the processing task while working with other application, which is less processing demanded, such as word processing, spreadsheet, etc. Those applications do not occupy the microprocessor fully. Thus, the processor may only be utilised less than 20 percent at most of the time, especially with a high performance microprocessor. A user can work on the word-processing without any trade off in performance, and use the rest 80 percent of available computer resources for the MSS processing. When the computer is only handling the MSS processing task, all available computer resources will be occupied. Furthermore, the MSS window resources are free, means, a user can post any command message while the other thread is processing the problem. Termination or status showing requests can be entertained. All these are impossible with the FDM system. Further improvement can be made to make true parallel processing a reality. To achieve such, a lot of effort is still required especially in the processing design.

The OOP is used in the MSS processor development. All the entities are handled by objects and classes. The use of OOP makes the development process more systematic. With the OOP, the expansion of the system to cope with different elements are made easy. Minimum effort or none may be required to add any new elements into MSS processor. To achieve such, a detail design of base classes and supportive classes is required and standards of element objects creation should be followed. Standardisation will have to be the next steps, which create standards to be followed in developing any numerical elements. The author revealed the possibility of developing a standard format with object-oriented design to deal with numerical methods, especially the FD method.

This will lead to a standard way of providing and storing information to and from the numerical method. This will make the FD development more efficient and may outrun the FE method.

The MSS processor has been used as an experimental platform to design, develop and test on new elements. Two new structure elements were developed. The MSS can execute either model implementation. The new models are more flexible and more suitable to be general-purpose models compared to the Witmer *et al.*'s model, which is implemented in the FDM. New contact-impact methods have developed, and free to be used with the present MSS processor. A new multipurpose revolute joint element is included, where resistive moment, rotary limits, locking and damping effects are made possible.

As a whole, the MSS processor gains a great improvement in both the implementation and the element applications. The user has more control during the processing and updated information is shown. Improvements of the MSS processors can be made in the future, such as:

- Standardise the OOP application in all element definitions
- Parallel processing
- Dynamic memory handling with swap files
- Better interface, which shows the current energies level and etc
- Continue processing of previous interrupted processing tasks (by force termination)

## 4.7 MSS POST-PROCESSOR

The MSS post-processor is called to view the result yielded from the MSS processor. The MSS processor stores the predicted responses and information into the MSS output data file. The data file will be retrieved and the data will be interpreted and displayed on the screen by the MSS post-processor. Most commonly, the transient responses of the structure are the main concern for a dynamic problem.

The MSS processor is having general features like animating the transient response, showing individual transient response at a specific time step. The MSS post-processor can generate user customised text-based data file, where the data is collected in different way according to the user request.

### 4.7.1 FUNCTIONS

The main function of the MSS post-processor is to help the user to "read" the output of the processed data from the MSS processor. The data is interpreted and presented in graphical form. Furthermore, different arrangements of the data collection are available, which the MSS post-processor generates special request data files. Any spreadsheet, such as Microsoft Excel can translate the text-based data for further analysis.

### 4.7.2 SYSTEM AND DATA FLOW

Figure 4-14 shows the system and data flow of the MSS post-processor. The main activities of the post-processor are placed under the CMSSPostDoc, a document class. Two view classes are involved, namely CMSSPostView and CMSSPostBotView. They are used to control the display output of two separate view panes. These two views are under the control of the CMainFrame, where the window split into two, top major and bottom minor displays.

The MSS post-processor reads the yielded data from the MSS processor-generated binary data output file. Data will be organised and displayed in graphical form. The graphical display will be handled by the CMSSPostView class while the CMSSPostBotView will display some information in numerical form.

The MSS post-processor is also capable of generate additional text-based data files, where the user will have the chance to customise the contents of the data file. They are called Special Request Output Data Files.

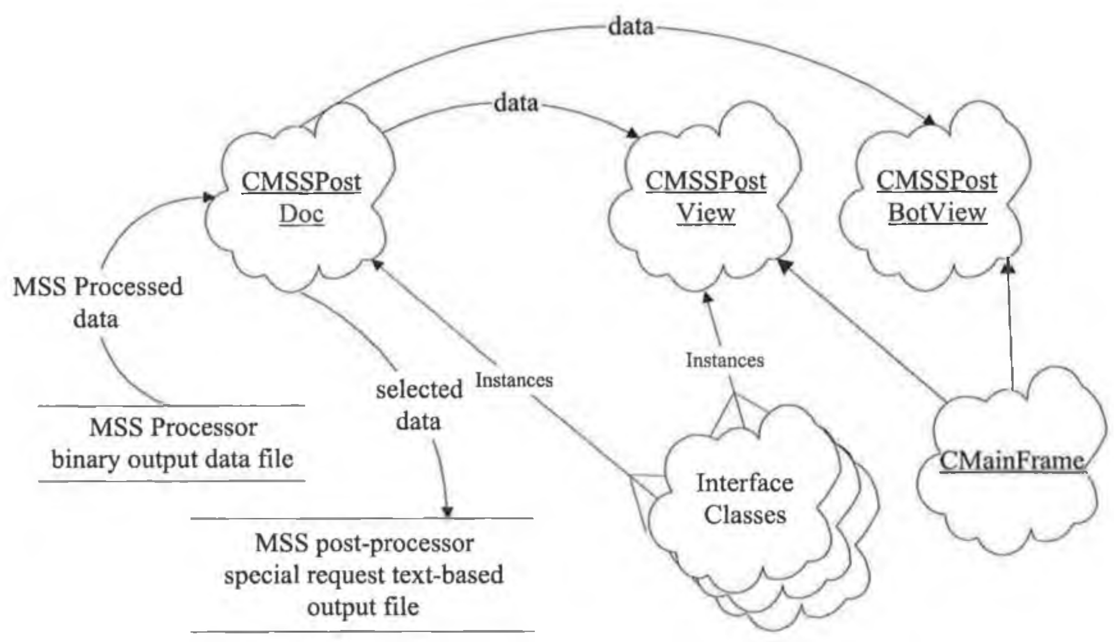


Figure 4-14: MSSPost system and data flow

## 4.7.3 CLASSES OVERVIEW AND INTERACTIONS

Figure 4-15 shows the class organisation of the MSS post-processor. The CMSSPostApp is the main executing entrance, where the application window is defined. The CMSSPostDoc, CMSSPostView and CMainFrame classes are bound within the CMSSPostApp.

The MSS post-processor used SDI with two panes. All panes are under the handling of the CMainFrame class. Each pane has a corresponding View class, namely CMSSPostView and CMSSPostBotView. Several interface classes are incorporated to handle interface between the user and the post-processor. All interface classes of the MSS Post-processor are listed under section B.10 of Appendix B.

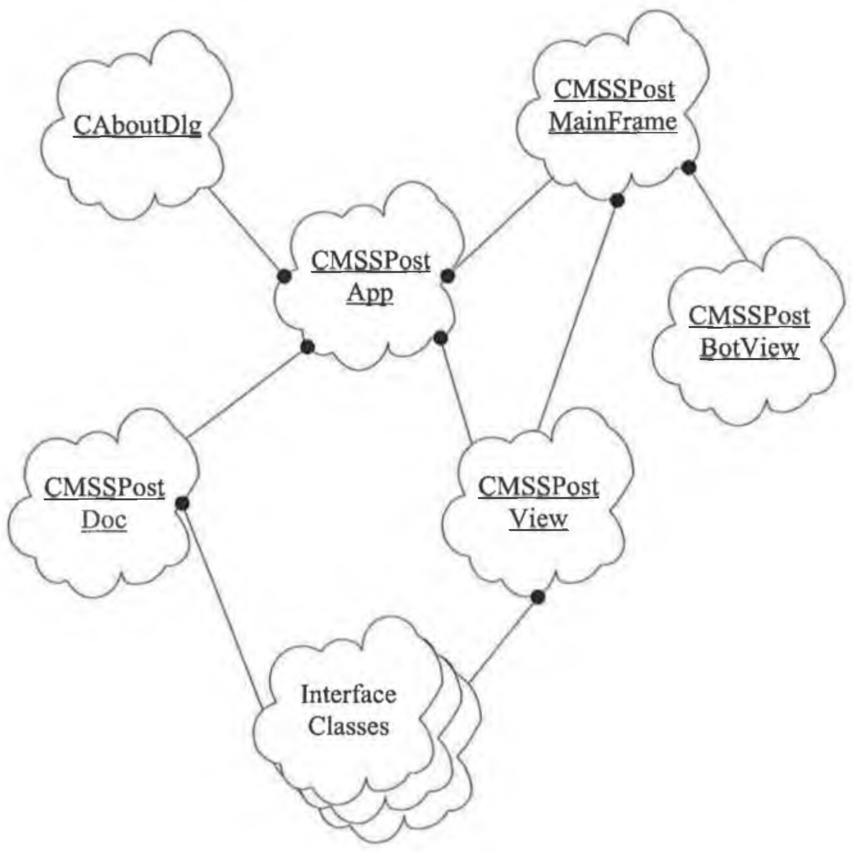


Figure 4-15: MSSPost classes organisation

### 4.7.4 MENU DEFINITIONS

The MSS pre-processor requires only one menu system. The MSS processor is using the SDI thus no additional main menu system is defined, which is similar to the MSS processor. Details of menu definitions can be found in B.11 of Appendix B.

The main functions of the MSS post-processor are located under *Track* and *View* menu. Under *Track*, all of the menu items are used to generate special request data files. While, under *View*, the menu items are selected to customise the display on the MSS post-processor window.

## 4.7.5 INPUT-OUTPUT FILE SYSTEMS

The MSS post-processor read MSS processor output data file. It generates customised output data files, which is called Special Request Data File in general. The first file system is in binary form with "otp" as extension, while the later is in text mode. The default extension of the Special Request Data File is "pox".

### A. MSS Processor Output Data File

The MSS Processor Output Data File is in binary form. It collects all the predicted transient responses as well as other corresponding data, which were created by the MSS Processor. The data will be *Serialized* by the MSSPostDoc class and stored into computer memory. Detail description of the file system is illustrated under section of Appendix B.

### B. Special Request Output Data File

A user can request more than one text-based output data file from the MSS post-processor. The special request data file consists of three types. They are different in collection of data. The first is collection all data about a node, second about a link, and the last one is all nodes and links data at a particular time step.

#### B.I Data About a Node

To generate special request data file about a node, a user has to select *Track – Node* in the main menu, which carries command message of *ID_TRACK_NODEVELOCITY*. The CMSSPostDoc will capture the message and routes it to OnTrackNodevelocity member function. The dialog box as shown in Figure 4-16 will be created, which is handled by the CDlgTcNdTel class.

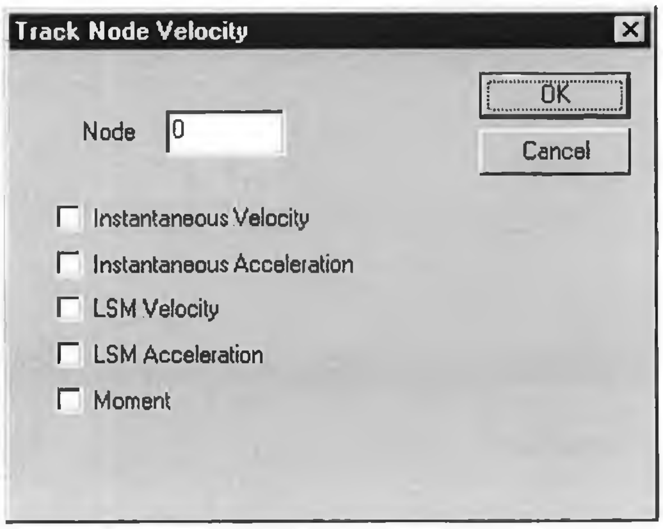


Figure 4-16: MSSPost track node dialog box

The user will be asked to provide the nodal reference number as well as to select the node attributes to be included. The displacement of the node is included by default. Two types of velocity and acceleration are shown, namely, instantaneous and the LSM. The first type shows the instantaneous velocity and acceleration, the velocity and acceleration at the very small size time step, which are calculated by the MSS Processor. While the LSM gives effective velocity and acceleration. The author used least squared method to predict the effective velocity and acceleration, which the formulation can be found in the following chapter. The LSM needs a parameter, *n*, which can be set in the MSS post-processor. The user has the choice to select any combination of the attributes shown in the above dialog box. Then a file-handling interface is created where a user can provide the target file name and its path. Standard heading will be placed as the beginning of the data file, which includes the source of the data (the MSS processor output data file), time and date generated, node reference number, and time step size. The nodal information at all time steps will be stored. A description about the data arrangement will be placed before series of data. The data contents will be very much depending on the option that set in the above dialog box. The following coding shows the result of node 19 from the tapered cantilever beam problem, where all the options in the above dialog box are selected.

```
WSV MSSPost exported data text file
by MSS Post-processor version 1.2, copyright(R) Jan 1999
---------------------------------
Base MSS Post-processor file: D:\WongSV\Ph.D. Project\VC project\MSS
Data\compare\taper.otp
Generated on Wednesday, November 03, 1999  at 16:30:13

Node: 19    dt:, 2.5e-006

Itrvl, TS, x, y, ins(Vx),ins(Vy), ins(Gx),ins(Gy), lsm(Vx),lsm(Vy),
lsm(Gx),lsm(Gy), moment
0, 0, 0,0.254, 34.3895,0, 0,0, 0,0, 0,0, 0
1, 200, 0.019843,0.25031, 35.9079,-12.8956, -48683.6,-21582.5, 0,0, 0,0, 0
2, 400, 0.0351738,0.243029, 26.1524,-14.8343, -2035.76,504.861, 0,0, 0,0, 0
3, 600, 0.0469672,0.235605, 21.7249,-15.334, -6386,-7142.07, 23.7837,-
13.8268, 0,0, 0
4, 800, 0.0569797,0.228061, 18.2814,-14.2461, -4429.25,2214.87, 19.6772,-
14.3602, 0,0, 0
5, 1000, 0.0656127,0.220845, 16.015,-13.9072, -862.435,609.676, 16.8607,-
14.007, 0,0, 0
6, 1200, 0.0732086,0.214108, 14.1016,-12.6329, -795.789,763.782, 14.8949,-
13.3795, -4008.22,849.454, 0
7, 1400, 0.0800982,0.207497, 13.1904,-13.6321, -2845.79,2083.37, 13.3808,-
12.6419, -2984.86,1000.39, 0
8, 1600, 0.0863601,0.201053, 11.8585,-11.5126, -1448.75,7759.89, 12.1962,-
12.1272, -2330.23,864.353, 0
9, 1800, 0.0920615,0.195622, 11.004,-10.2985, -369.235,-1381.8, 11.226,-
11.8064, -1900.18,632.039, 0
10, 2000, 0.0974069,0.190233, 10.268,-10.9991, -738.403,202.611, 10.4039,-
11.5762, -1591.68,458.334, 0
.....................
```

Coding 4-2: Sample contents of MSSPost Special Request Data File about a node

## B.II Data About a Link

Collection of transient data about a link can be placed into Special Request Data File. To generate such data file, *Track – Link* has to be selected in the main menu. The CMSSPostDoc will capture the command message *ID_TRACK_LINK* and routes it to the OnTrackLink member function. A dialog box will be created, which is handled by the CDlgTrackLk class. The link reference number will have to be provided. At the present moment, only the bending moments at both nodes of the link are included. A typical data file is shown as the following.

```
WSV MSSPost exported data text file
by MSS Post-processor version 1.2, copyright(R) Jan 1999
---------------------------------
Base MSS Post-processor file: D:\WongSV\Ph.D. Project\VC project\MSS
Data\compare\taper.otp
Generated on Wednesday, November 03, 1999  at 19:30:10

Link: 1    dt: 2.5e-006

Itrvl, TS, GMmtN1, GMmtN2
0, 0, 0, 0
1, 200, -1.36756e-006, -2.26321e-005
2, 400, -0.0202661, -0.0547814
3, 600, -0.115587, -0.049743
4, 800, -0.167752, 0.00420111
5, 1000, 0.111018, -0.0630726
6, 1200, 0.28365, -0.146315

..............................
..............................

107, 21400, -0.069125, 0.0941315
108, 21600, -0.0774098, 0.0877743
109, 21800, -0.0910145, 0.0902647
110, 22000, -0.107161, 0.0917342
111, 22200, -0.121909, 0.0858481
112, 22400, -0.115604, 0.0760914
113, 22600, -0.0909631, 0.0683508
114, 22800, -0.0739132, 0.0563219
115, 23000, -0.0691979, 0.0470775
116, 23200, -0.0729867, 0.0325674
117, 23400, -0.056493, 0.0251503
118, 23600, -0.0398679, 0.00644657
119, 23800, -0.0213218, -0.0102382
120, 24000, -0.000340783, -0.0215857
```

Coding 4-3: Sample contents of MSSPost Special Request Data File about a link

## B.III Data at a Time Step

Collection of transient data of all nodes at a particular time step can be placed into Special Request Data File. To generate such data file, *Track – Time* has to be selected in the main menu. The CMSSPostDoc will capture the command message *ID_TRACK_TIME* and routes it to the OnTrackTime member function. A dialog box will be created, which is handled by the CDlgTcTime class. The time step interval number will have to be provided. At the current stage, only the nodal displacement and bending moments are included.

## 4.7.6 FUNCTIONS IMPLEMENTATION

The MSS post-processor has several display features. Animate the transient response is one of the most important. A user can select a particular time step to be display. In addition, a user can also display transient responses at several time steps.

### A. Animate

To start animating of the transient responses, *View – Animate* menu item has to be selected. Upon the selection, a command messages *ID_VIEW_ANIMATE* will be posted. The CMSSPostView class will capture the message and route to the OnViewAnimate member function. The following Nassi-Schneiderman chart shows the implementation of the OnViewAnimate function.

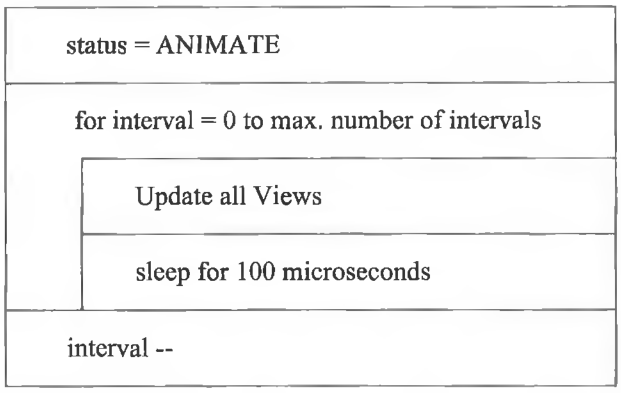

status = ANIMATE
for interval = 0 to max. number of intervals
Update all Views
sleep for 100 microseconds
interval --

Figure 4-17: Animate function implementation

The CMSSPostView::OnDraw function will use the *interval* number to display the transient response at the time step interval. Each iteration will increase the *interval* value and each update of the Views will display the transient response at the new time step interval.

### B. Freeze

Freeze function will only display the transient data at a single time step interval. To achieve such, *View – Freeze* menu item has to be selected. Upon the selection, a command messages *ID_VIEW_FREEZE* will be posted. The CMSSPostView class will capture the message and route to the OnViewFreeze member function. A dialog box controlled by the CFreezeDlg class will be created and the user will have to provide the number of time step intervals. This value will be used when updating the view later at the end of the function implementation.

### C. Overall

Transient responses of several time step intervals can be displayed simultaneously for comparison purpose. Firstly, *View – Overall* menu item has to be selected. Upon the selection, a command messages *ID_VIEW_OVERALL* will be posted. The CMSSPostView class will capture the message and route to the OnViewOverall member function. A dialog box controlled by CDlgViewOverA class will be created to let the user enter the time step intervals and select some options. Figure 4-18 shows the display of the dialog box.

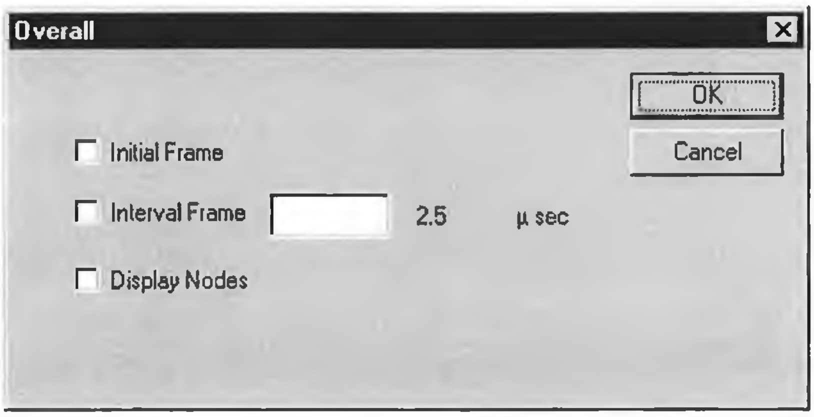


Figure 4-18: Overall display dialog box

Several options are available to the user. They are initial structure display, multiple intervals display, and show nodes. The user will have to provide the desired number of intervals to be shown, which are separated by comas " , ".

Upon finishing of the dialog box, all the information will be stored into the local variables and update views message is then issued. In the OnDraw function, the display of the view will be altered accordingly.

### 4.7.7 DISPLAYS

To show the display of the MSS post-processor, the problem of the tapered cantilever beam problem is used. Upon loading of the MSS Processor output data file, the overall profile of the transient response will be shown. The Figure 4-19 shows a typical display after reading the data file. The initial structure is shown as solid line and the displacements of all nodes are displayed.

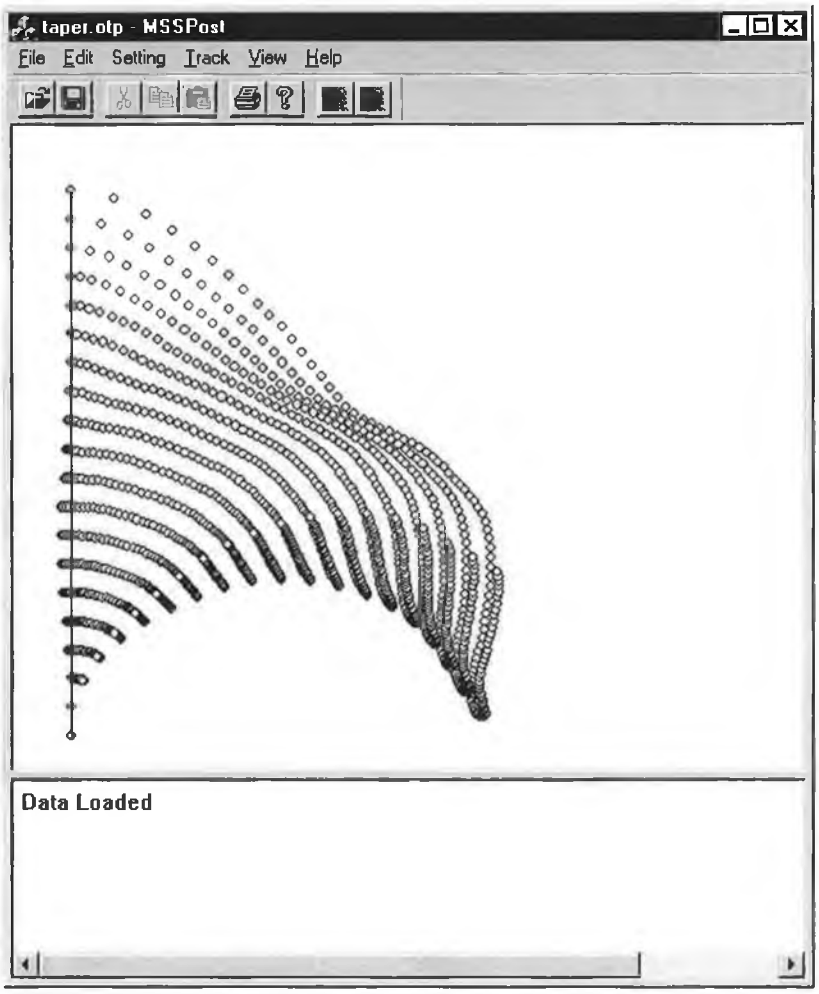
Figure 4-20 shows a display example of a freezed transient display at time step interval of 30, which is equivalent to 6000 time steps and 0.015 s. The overall kinetic energy, elastic energy and plastic energy is shown at the bottom frame. A customised overall display is shown in Figure 4-21, with initial frame and interval frame options selected with intervals of 10, 20, 30, 40, 60, 80, 100, and 120 (entered in the dialog box as shown in Figure 4-18).

### 4.7.8 DISCUSSIONS AND SUMMARIES

The MSS post-processor provides crucial functions to help the user to visualise the result of the MSS processor. The MSS post-processor is capable of animating the transient responses of the solved problem as well as displaying a single frame at a specific time step interval. The FDM has already achieved up to this stage.

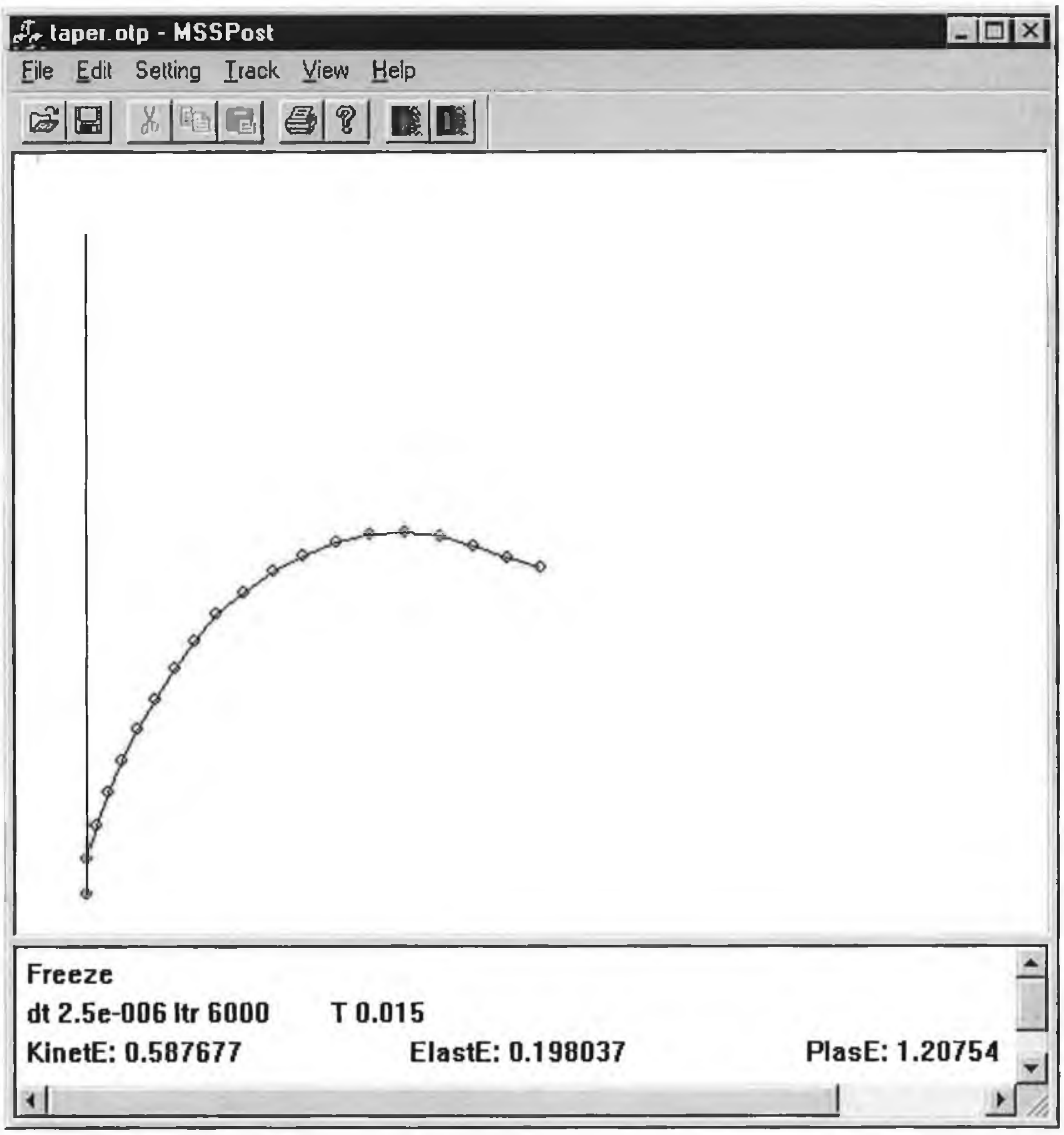
MSS post-processor uses different panes to display some numerical information, such as the time steps, energies level, et cetera. The user can customise the overall display of the solution in MSS post-processor, where response of several intervals can be display simultaneously. Special Request Output Data File can be generated upon the needs of the user. Several options and customisation are available with the MSS post-processor. The data file can later be imported to any commercial available spreadsheet or any numerical data processing application software, such as Microsoft Excel, for further interpretation.

The MSS post-processor has features, which could be added. Several improvements are suggested, they are contour display of certain quantities, more selection in customising Special Request Data File, automate graphs generation inside the MSS post-processor and more userfriendly.



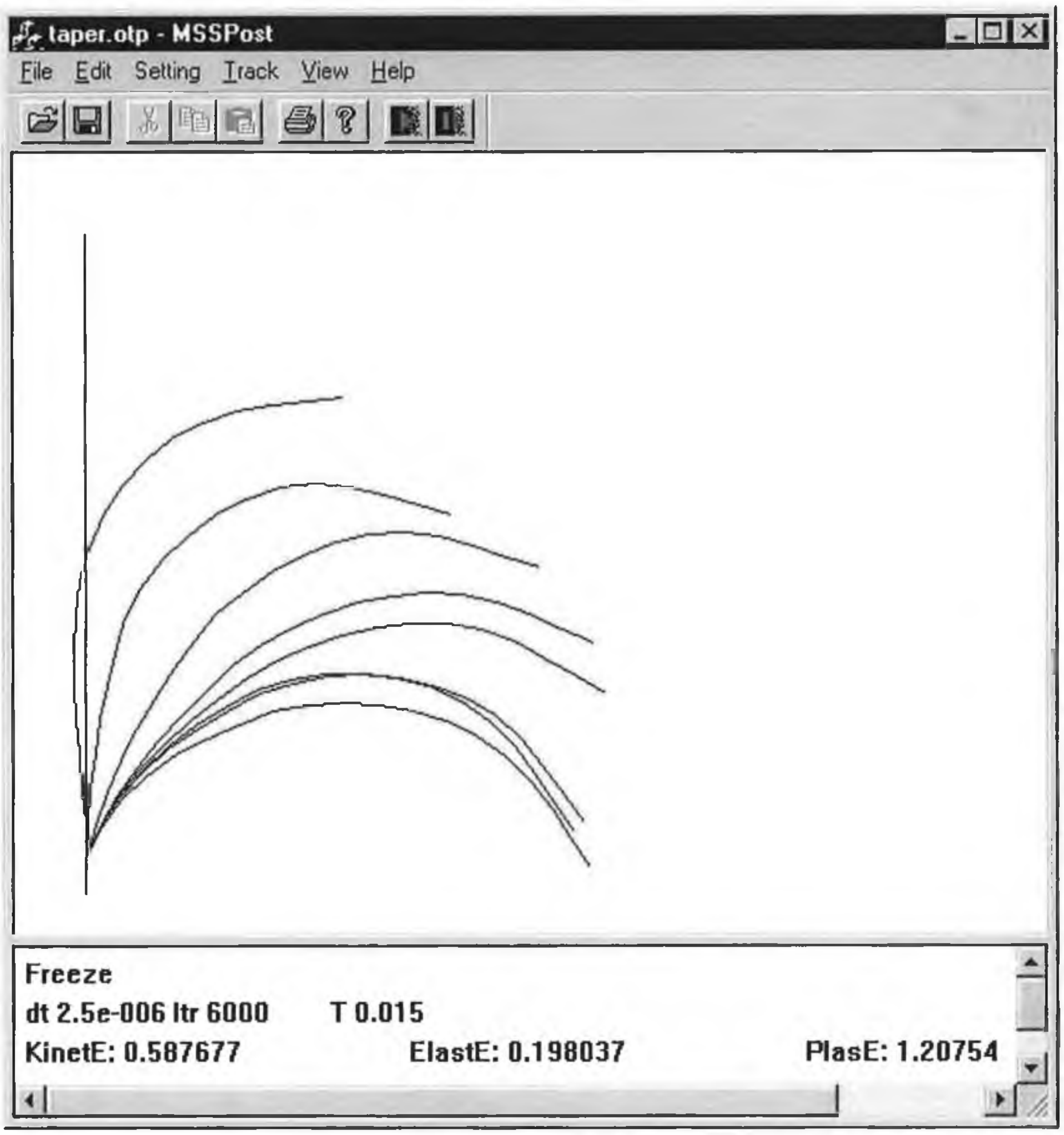
Data Loaded

Figure 4-19: Typical display of MSSPost after reading processed data file



```
Freeze
dt 2.5e-006 Itr 6000        T 0.015
KinetE: 0.587677            ElastE: 0.198037            PlasE: 1.20754
```

Figure 4-20: Typical freeze transient response display of MSSPost



```
Freeze
dt 2.5e-006 Itr 6000        T 0.015
KinetE: 0.587677            ElastE: 0.198037            PlasE: 1.20754
```

Figure 4-21: Typical customised overall display of MSSPost

## 4.8 MSS APPLICATION METHODOLOGY

The MSS application methodology is very similar to the FDM. Figure 3-24 can be used to illustrate the methodology.

# Chapter 5: FORMULATIONS & THEORIES

## 5.1 INTRODUCTION

Numerical formulations incorporated in both the FDM and the MSS, and the respective theory behind these are presented in this chapter. Validations are made with exact solutions and experimental results from published literatures. The formulations include the structural elements, time (temporal) integration scheme, material modelling, boundary conditions, contact algorithms, and revolute joint formulations. Besides that, the stability of the system, from both the structural and revolute joint elements is discussed.

## 5.2 FDM STRUCTURAL ELEMENT FORMULATION

The FDM system incorporates FD model of Witmer *et al.*'s [1] and Hashmi *et al.* [28], which is a general method developed to predict the response and permanent plastic deformation of shell structures under dynamic loading conditions. The model is a 1-D model with infinite shear rigidity in 2-D space and is capable of handling large deformation and different material properties. The model itself is based on a generalised numerical method. The effects of rotary inertia and transverse shear deformations are ignored in this model. Figure 5-1 illustrates the lumped-parameter numerical model, which shows two connected link elements.

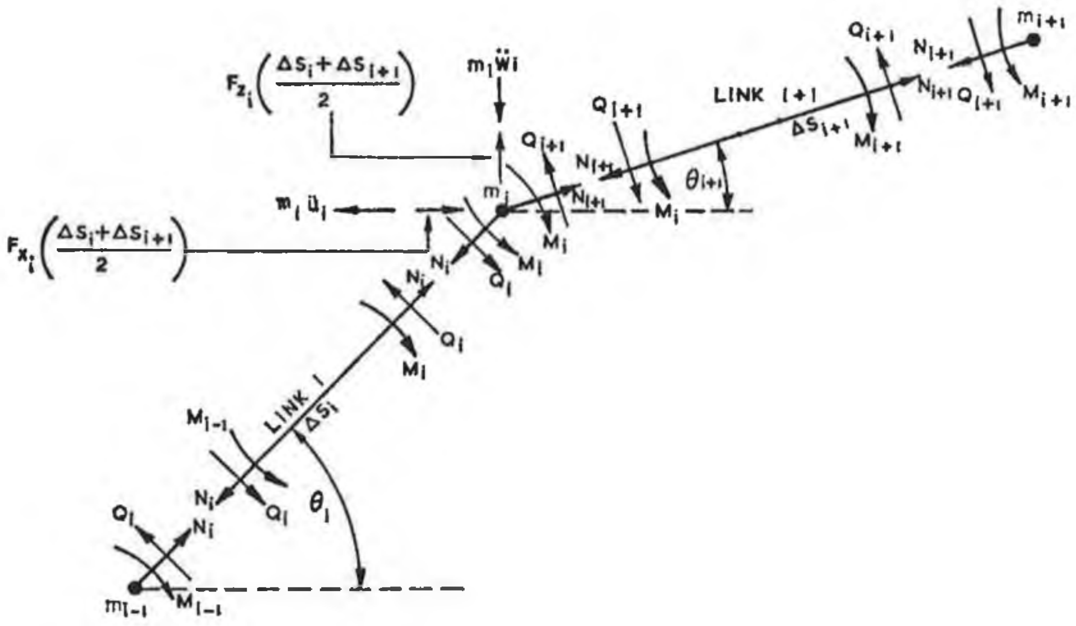


Figure 5-1: Numerical model for the structure

### 5.2.1 EQUATIONS OF MOTION

The finite difference equations of dynamic equilibrium of the *i*th node connecting *i-1*th and *i*th elements are

$$N_{i+1}\cos\theta_{i+1} - N_i\cos\theta_i - Q_{i+1}\sin\theta_{i+1} + Q_i\sin\theta_i + F_{x_i}\left(\frac{\Delta s_i + \Delta s_{i+1}}{2}\right) - m_i\ddot{u}_i = 0$$

(5-1)

$$N_{i+1}\sin\theta_{i+1} - N_i\sin\theta_i + Q_{i+1}\cos\theta_{i+1} + Q_i\cos\theta_i + F_{z_i}\left(\frac{\Delta s_i + \Delta s_{i+1}}{2}\right) - m_i\ddot{w}_i = 0$$

(5-2)

where $N_i$ is the axial force acting on the *i*th element. The other internal forces are the transverse shear force, $Q_i$, and the bending moment, $M_i$. $F_{x_i}$ and $F_{z_i}$ are the external forces acting in x and z direction of the global co-ordinate system, which are functions of structure location, *s*. $\Delta s_i$ is the length of the element. The internal and external forces causes the lumped mass of node *i*th to move with acceleration $\ddot{u}_i$ in the *x* direction and $\ddot{w}_i$ in the *z* direction.

The relation between the shear force and the bending moment is expressed as,

$$M_i - M_{i-1} - Q_i \Delta s_i = 0 \tag{5-3}$$

## 5.2.2 CROSS SECTION IDEALISATION

Cross section idealisation is used to determine the internal forces and moments. The FDM formulation idealises the actual cross section as consisting of *n* discrete, evenly spaced, equal-area layers of material that can carry normal stresses. These layers are considered to be separated by material that cannot carry normal stresses but has infinite shear rigidity. Figure 5-2 shows the idealised thickness model.

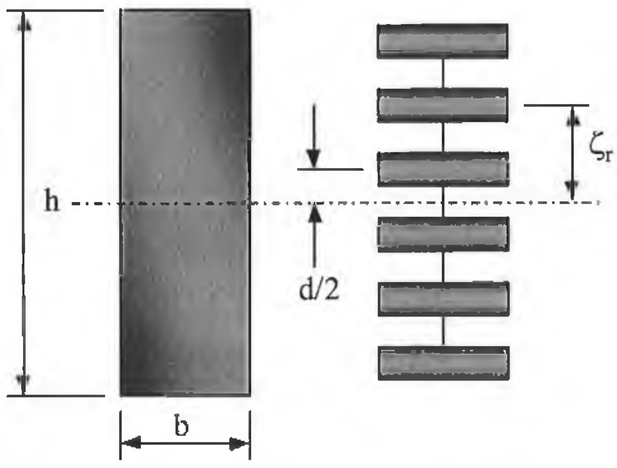


Figure 5-2: Thickness model for the structure

The stress and strain in the structure can be defined by the individual normal stresses in the *n* layers, invoking the assumption that plane sections remain plane throughout the response. The value of *d*, the space between layers and area of each layer, *a*, are obtained with the following equations.

$$d = h / n \tag{5-4}$$

$$a = bh / n \tag{5-5}$$

## 5.2.3 STRAIN-DISPLACEMENT RELATIONS

The strain in the *r*th layer located at a distance $\zeta_r$ above the centroidal axis at mass point *i* may be expressed approximately as

$$\varepsilon_i^r = \frac{\Delta s_i - \Delta s_{i,0}}{\Delta s_{i,0}} - \zeta_r \frac{\Delta\theta_i - \Delta\theta_{i,0}}{\frac{1}{2}\left(\Delta s_{i+1,0} + \Delta s_{i,0}\right)} \tag{5-6}$$

where $\Delta s_i$ is the length of the link *i* and $\Delta\theta_i$ is the angle between the links *i* and *i*+1. The index 0 refers to the initial position. The first term in Equation (5-6) is the axial strain in the *i*th link, and the second term represents the bending strain contribution evaluated at the *i*th mass node. For sufficiently small angles $\Delta\theta_i$, it can be approximated with the following equation.

$$\begin{aligned} \Delta\theta_i &\approx \sin\Delta\theta_i \\ &= \sin(\theta_{i+1} - \theta_i) \\ &= \sin\theta_{i+1}\cos\theta_i - \cos\theta_{i+1}\sin\theta_i \end{aligned} \tag{5-7}$$

The change of strain in the *r*th layer at time $t_{j+1}$ can be determined with the following equation.

$$\Delta\varepsilon_{i,j+1}^r = \varepsilon_{i,j+1}^r - \varepsilon_{i,j}^r \tag{5-8}$$

Associated stresses in the layers can then be obtained from appropriate stress-strain relations of the material selected. Once these stresses are found, the axial force and moment at each mass-point station may be computed with Equations (5-9) and (5-10).

$$N_{i,j+1} = a \sum_{k=-n/2}^{n/2} \sigma_{i,j+1}^k \tag{5-9}$$

$$M_{i,j+1} = a \sum_{k=-n/2}^{n/2} \sigma_{i,j+1}^k \zeta_k \tag{5-10}$$

### 5.2.4 DISCUSSIONS AND SUMMARIES

The numerical method proposed by Hashmi *et al.* [28] is simple both in terms of understanding and implementation. It can be considered as an Eulerian hydrocode. The whole implementation is simple and can easily be adapted for different problems. This was proven with a great number of impact dynamic problems, which were solved with models of similar kind throughout these years. The numerical results are well correlated with experimental results. On the whole, the method is good and simple to implement.

The element formulations are based on previous predefined connected link. This can be seen from the strain and motion equations. The whole structure is considered as one and each element is not "independent" in term of implementation as well as formulations of the processing task. This puts a constraint to the development of a general, multi-purpose processor. This has been a typical drawback of many classical FD methods. In other words, no provision in the FD formulations that can link all of them together. The FE method provides a better solution for this purpose.

The other significant drawback of the method is its modelling constraints. The method is only capable of handling simple structure where extra extension is not made. The problem has to be represented by either an opened loop model or a closed loop. Figure 5-3 below shows example of opened-loop, extension, and closed-loop structures.

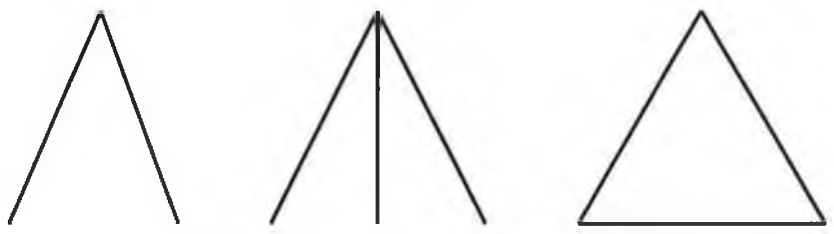


Figure 5-3: Opened-loop, extension, and closed-loop structure examples.

From these two drawbacks, the FD methods of similar type are not suitable to be incorporated into a general-purpose solver.

## 5.3 MSS STRUCTURAL ELEMENTS FORMULATION

Both numerical formulations by Witmer *et al.* and Hashmi *et al.* are lacking of flexibility in representing the structures and problems. General finite difference models for structural impact simulation described above are not suitable for use without the presence of the developers. The models are designed to represent simple open-ended structures for which the boundary conditions have to be clearly specified for different problem representations.

This section presents a new 1-D finite difference model in predicting dynamic responses of structures in 2-D space from elastic to plastic deformation. Central difference method for time integration is incorporated into the model. The model can handle different shapes including T-joint, even connecting a node of a link to several links. The model enables definition of different constraints in a structure, from full to partial hinge arrangement. The nature of the model makes the computer execution and implementation easier and suitable for different structural orientations and shapes with different material properties simultaneously.

### 5.3.1 EQUATIONS OF MOTION

A structure subjected to external forces can be generalised with Equation (5-11) when considering dynamic equilibrium.

$$\frac{\partial \mathbf{F}_{\text{external}}}{\partial s} - \frac{\partial \mathbf{F}_{\text{internal}}}{\partial s} = \mathbf{MA} \tag{5-11}$$

where **M** and **A** are the overall mass and overall acceleration of the structure. Because the model of interest in this section is a 1-D finite difference model, the partial differential Equation (5-11) is differentiated with *s*, the length of the structure. $\mathbf{F}_{\text{external}}$ is the external applied load on the structure. $\mathbf{F}_{\text{internal}}$ is the internal reaction force from its orientation and deformation. The moment equilibrium about axis-w can be expressed as:

$$\frac{\partial M}{\partial s} - Q = 0 \tag{5-12}$$

where $M$ is the moment, which includes the externally applied moment and moment caused by the bending of the structure. The applied moment can be calculated and then considered in terms of one of the external forces in Equation (5-11). For simplicity of the computing efforts, all the internal bending moments and externally applied moment have been incorporated into Equation (5-12). The term $Q$ is the shear force, which is taken into consideration as a part of $\mathbf{F}_{\text{internal}}$ in Equation (5-11).

The structure is idealised into a mass-link system and is divided into several finite small elements. The element in this case is a link and consists of two nodes. The mass of the link is lumped into corresponding end nodes. As a result, a mass point (node) is lumped from all its-connected links with the following equation:

$$m_{\text{node}\,a} = \sum_{i=\{\text{connected links}\}} \frac{(\rho A \Delta s)_{\text{link}\,i}}{2} \tag{5-13}$$

where $\rho$ is the density of the material, $A$ is the area of the cross-section and $\Delta s$ is the length of the link.

The link is weightless and transmits axial forces, shear forces and bending moment. The position of a node in the model identifies the position of an element of the actual structure. The following formula is required to express the equilibrium relations of the nodes, thus obtain the corresponding nodal displacement.

$$\mathbf{F}_{\text{external}}(\mathbf{u}, \mathbf{v}, \mathbf{a}, t) - \mathbf{F}_{\text{internal}}(\mathbf{u}, \mathbf{v}, \mathbf{a}, t) = \mathbf{MA} \tag{5-14}$$

where $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{a}$ consist of all the corresponding local tensors of displacement, velocity and acceleration respectively. $\mathbf{M}$ is the lumped mass array and $\mathbf{F}_{\text{external}}$, $\mathbf{F}_{\text{internal}}$ and $\mathbf{A}$ are the arrays containing external applied load tensors, internal force tensors and resultant acceleration tensors respectively.

An array is referring to a 1-D matrix, where each element of the array representing the data pertaining to a particular node. The time instance, $t$, is included explicitly in

Equation (5-14). The multiplication operation between the arrays, **M** and **A**, is a direct multiplication, as the following:

$$\begin{aligned}
&\mathbf{M} = [m_1 \quad m_2 \quad .. \quad m_n] \\
&\mathbf{A} = [\mathbf{a}_1 \quad \mathbf{a}_2 \quad .. \quad \mathbf{a}_n] \\
&\text{then} \\
&\mathbf{MA} = [m_1 \times \mathbf{a}_1 \quad m_2 \times \mathbf{a}_2 \quad .. \quad m_n \times \mathbf{a}_n]
\end{aligned} \tag{5-15}$$

With this arrangement, generation of a huge lumped mass matrix is avoided.

## 5.3.2 EXTERNAL AND INTERNAL FORCES

The $\mathbf{F}_{\text{external}}$ consists of all the possible external forces, ranging from constant loading, regardless of any other dynamic factor, to a function of time and location. In addition, external damping, as a function of velocity, can also be applied. Contact reaction of the structure when collision with other structure(s) can be included.

The $\mathbf{F}_{\text{internal}}$ is a collection of all the structural responses due to the reaction from the deformation of the structure. For the 1-D finite difference model, the internal forces comprise of axial forces and shear forces normal to the centroidal axis of the structures, which arise from bending moments. All the corresponding internal and external forces are then summed up at a node.

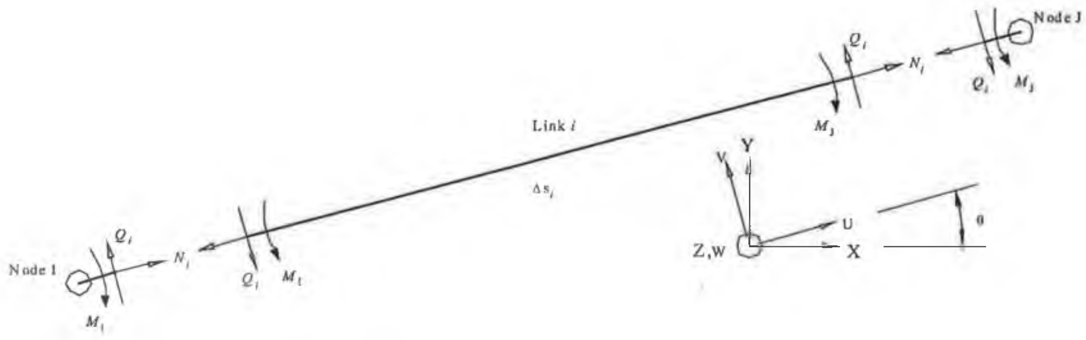


Figure 5-4: Link i internal forces

### 5.3.3 ELEMENT COMPONENTS

Figure 5-4 shows a typical link with internal force arrangement, and contributions towards its member nodes. The nomenclatures for the link and the corresponding local co-ordinate system are shown. As can be seen an element consists of two nodes connected with a link, denoted as node-I and node-J. The link is weightless and infinitely rigid in bending. It only carries axial stress, denoted as $N_i$ in Figure 1. The bending moment and externally applied moment around node-I and node-J are denoted as $M_\text{I}$ and $M_\text{J}$ respectively. Equation (5-16) is formulated when considering Equation (5-12) and referring to an element.

$$M_\text{J} - M_\text{I} - Q_i \Delta s_i = 0 \tag{5-16}$$

To cope with 2-D structure problems, two 3x3-transformation matrices are used to connect the element local tensors with the global tensors. They are the ***traverse*** matrix and z-axis ***rotation*** matrix, as shown in Equations (5-17) and (5-18).

$$T_{xy}(dx, dy) = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \tag{5-17}$$

$$Rz(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5-18}$$

where *dx* and *dy* are the desired traverse magnitude in global x-axis, y-axis and z-axis respectively. As a normal practice to save computer memory resources, $T_{xy}$ is applied with *dx* and *dy* as the location of the node-I when evaluating an element *i*, thus the node-I is the origin of the local co-ordinate system. A rotation transformation is then performed, with $\theta$ as shown in Figure 5-4.

All the bending is concentrated at the nodes. All the internal forces act at the corresponding nodes and are stored in the $\mathbf{F}_\text{internal}$ array in Equation (5-14). External forces are applied to the corresponding nodes, and stored in the $\mathbf{F}_\text{external}$ array.

The contribution of internal forces of a link $i$ at node-I and node-J to the global $\mathbf{F}_{\text{internal}}$ are shown as the following equations.

$$\mathbf{F}^{a}_{\text{internal}}{}' = \mathbf{F}^{a}_{\text{internal}} + R_z(\theta)\begin{bmatrix} Q_i \\ N_i \\ 0 \end{bmatrix} \tag{5-19}$$

$$\mathbf{F}^{b}_{\text{internal}}{}' = \mathbf{F}^{b}_{\text{internal}} + R_z(\theta)\begin{bmatrix} -Q_i \\ -N_i \\ 0 \end{bmatrix} \tag{5-20}$$

where $\mathbf{F}^{a}_{\text{internal}}$ is the element $a$ of the $\mathbf{F}_{\text{internal}}$ array corresponding to global node $a$, the node-I of link $i$. $\mathbf{F}^{b}_{\text{internal}}$ is the element $b$ of the $\mathbf{F}_{\text{internal}}$ array corresponding to global node $b$, the node-J of link $i$. The superscripted "$'$" indicates the updated force vector.

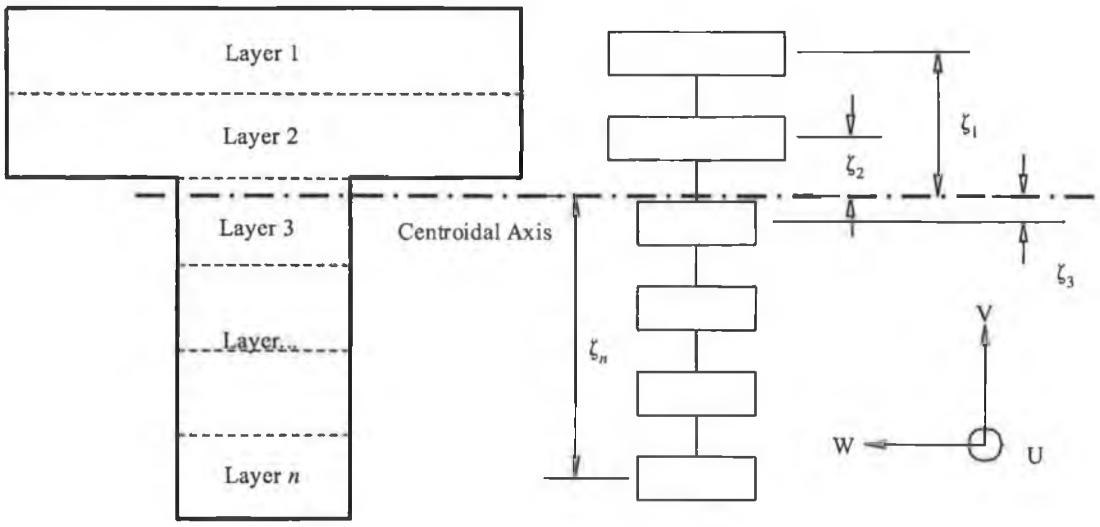


Figure 5-5: Cross-section example and corresponding idealised cross-section

### 5.3.4 CROSS-SECTION IDEALIZATION AND STRAIN-DISPLACEMENT RELATIONS

Because of the nature of the 2-D space modelling, only symmetrical bending is included. Asymmetrical or skew bending can only be modelled properly with moment around V-axis, besides the bending moment around W-axis. Thus, the cross-section of the link must have symmetry over plane U-V. Figure 5-5 shows a possible cross-section, for example. The cross-section of the structure is idealised as *n* discrete layers. The discrete layers carry normal stresses and are interconnected with material that cannot carry normal stresses but has infinite shear rigidity. The stress and strain in the structure can be defined by the individual normal stresses in *n* layers, invoking the assumption that plane sections remain plane throughout the response. Figure 5-5 also shows the idealisation of the cross-section. For normal practice, the V-axis has its origin aligned with the centroidal axis.

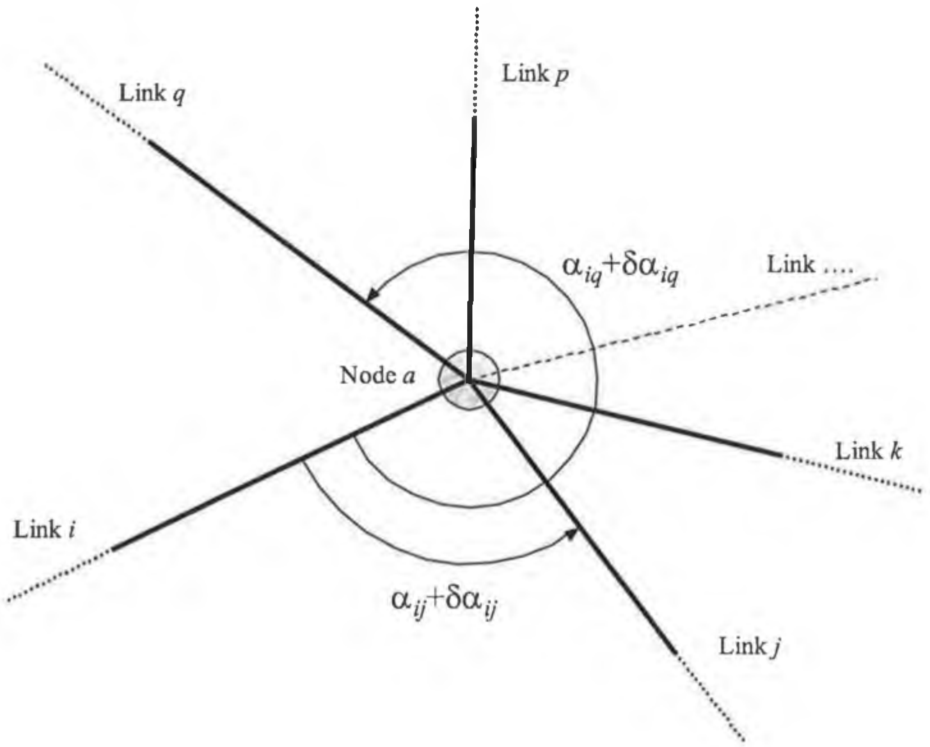

Link $q$
Link $p$
$\alpha_{iq}+\delta\alpha_{iq}$
Link ....
Node $a$
Link $k$
Link $i$
$\alpha_{ij}+\delta\alpha_{ij}$
Link $j$

Figure 5-6: Bending consideration with Bending-Division method

To handle bending between a link and its neighbouring links, two different models are suggested. The first with the assumption that the change of bending angle at a node with respect to a link *i*, which connects a number of links, equals to the superimposition of the change of angle at the node for the two neighbouring links with respect to link *i*. Figure 5-6 shows a link, link *i*, connected to a number of links through the common node *a* (which could be either node-I or node-J for each respective links). The two neighbouring links are denoted as link *j* and link *q* as shown in Figure 5-6. $\delta\alpha_{ij}$ and $\delta\alpha_{iq}$ are the changes of $\alpha_{ij}$ and $\alpha_{iq}$ at a particular time instant, *t*. The change of strain at layer *r* of node-J in link *i* from time *t* to time *t*+1 can be expressed with the following equation:

$$\begin{aligned}\delta\varepsilon^{r}_{i,\mathrm{J},t+1} &= \varepsilon^{r}_{i,J,t+1} - \varepsilon^{r}_{i,\mathrm{J},\mathrm{t}} \\ &= \frac{\delta\Delta s_{i,t+1}}{\Delta s_{i,0}} - \zeta_r\left(\frac{\delta\alpha_{ij,\mathrm{J},\mathrm{t+1}} + \delta\alpha_{iq,\mathrm{J},\mathrm{t+1}}}{2\Delta s_{i,0}}\right)\end{aligned} \tag{5-21}$$

where, $\zeta_r$ is the height of flange *r* from the centroidal axis. Similar arrangement is made for node-I as well.

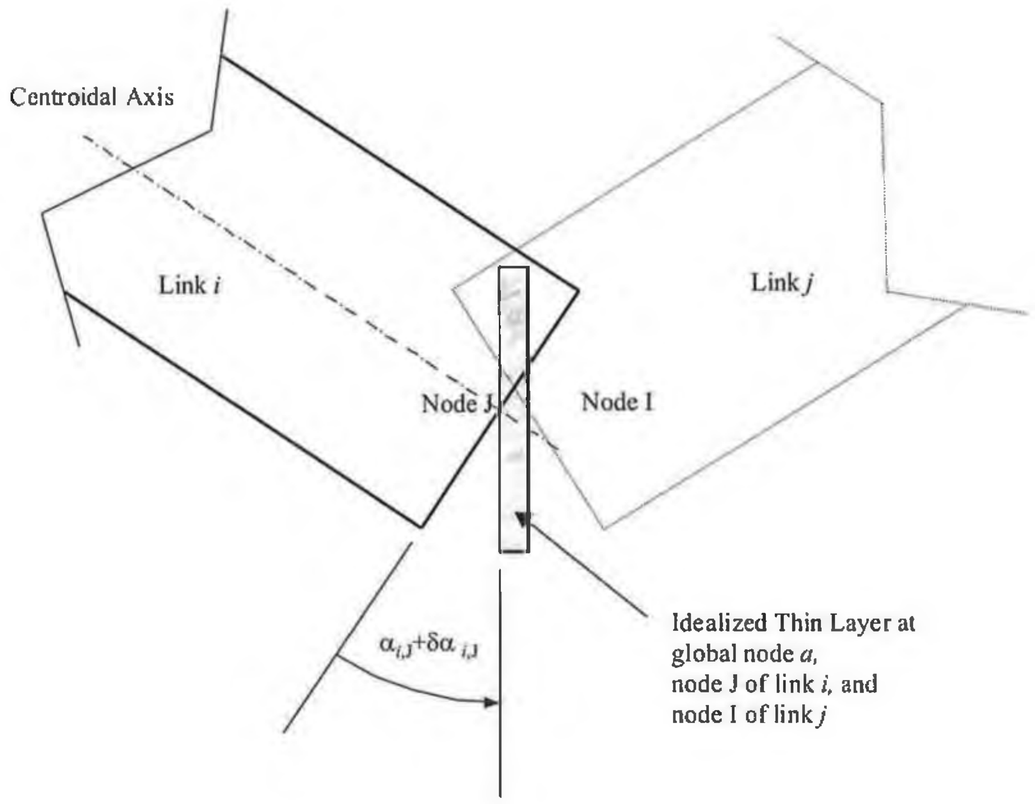

Centroidal Axis
Link *i*
Link *j*
Node J
Node I
$\alpha_{i,J}+\delta\alpha_{i,J}$
Idealized Thin Layer at global node *a*, node J of link *i*, and node I of link *j*

Figure 5-7: Bending consideration with Node-Layer method

The second model makes use of the cross-section and the lumped mass idealisation. Links are connected with an idealised thin and solid layer, which carries the mass of the node (refer to Equation (5-13)). The plane carries no normal stress and has infinite shear rigidity, thus no bending occurs on the plane. Figure 4 shows idealisation of link $i$ boundary at node-J. The change of strain is then expressed as Equations (5-22).

$$\delta\varepsilon^{r}_{i,\mathrm{J},t+1} = \frac{\delta\Delta_{i,t+1}}{\Delta s_{i,0}} - \zeta_r\left(\frac{2\delta\alpha_{i,\mathrm{J},t+1}}{\Delta s_{i,0}}\right) \tag{5-22}$$

$$\delta\alpha_{i,\mathrm{J},t+1} = \delta\beta_{a,t+1} - \delta\theta_{i,t+1} \tag{5-23}$$

where $\delta\beta_{a,t+1}$ is the angle change of the idealised layer's boundary at global node $a$ from time $t$ to time $t+1$. The $\delta\theta_{i,t+1}$ is the change of $\theta$ in Figure 5-4. The angular acceleration of $\beta_{a,t+1}$ is determined before hand with the following equilibrium equation.

$$\ddot{\beta}_{a,t+1} = \frac{\sum\limits_{i=\{\text{connected links}\}} M_{i,a,t}}{I_a} \tag{5-24}$$

$$I_a = \sum_{i=\{\text{connected links}\}} \sum_{r=1..n} \frac{m_i A_{r,i} \zeta^2_{r,i}}{2A_{c,i}} \tag{5-25}$$

where $M_{i,a,t}$ is the moment of the node $a$ contributed from the link $i$ at time $t$. It is based on assumption that no initial bending at time $t$=0. The node $a$ can be either local node-I or node-J. The $I_a$ is the representative moment of inertia of the corresponding idealisation of the cross-section. The term $n$ is the number of idealised layers, $A_{r,i}$ is the area of layer $r$ on the cross-section of link $i$ and $A_{c,i}$ is the total area of link $i$ cross-section. The term $m_i$ is the mass of the link $i$.

The first model is named as Bending-Division method and the latter as Node-Layer method. With both methods mentioned above, the associated layer stresses can be obtained from appropriate stress-strain relations. Once these stresses are found, the axial force and moment at each node corresponding to a link can be found with the equations below.

$$N_{i,t+1} = \sum_{r=1}^{n}(\sigma_{i,\mathrm{I},t+1}^{r} + \sigma_{i,\mathrm{J},t+1}^{r})\frac{A_r}{2} \tag{5-26}$$

$$M_{i,\mathrm{I/J},t+1} = \sum_{r=1}^{n}\sigma_{i,\mathrm{I/J},t+1}^{r}A_r\zeta_r \tag{5-27}$$

### 5.3.5 NUMERICAL EXAMPLES

For initial validation, a triangular frame subjected to impulsive load is chosen. Figure 5-8 shows the geometrical setting and loading condition. The triangular frame made from 0.5 inch (12.7mm) wide and 0.036 inch (0.9144mm) thick aluminium sheet metal. It was subjected to concentrated explosive impulse at one of its sides. I.C.I.-made nylon detonators, delivering an average impulse of 1.5x10-2 lb-sec, were used. The impulse was being obtained from the swing of a ballistic pendulum. Numerical and experimental results are provided in reference [28]. Similar initial conditions are provided and locations of applying the impulsive load with respect to time are monitored. Comparison between the two new models with the numerical model of Hashmi *et al.* is made.

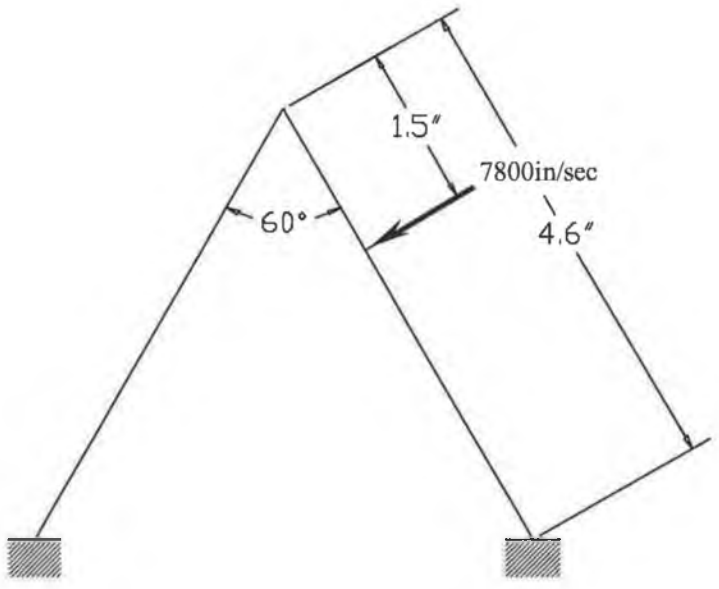


Figure 5-8:Triangle frame geometrical setting and loading condition

Table 5-1 shows the detail comparison of results. Figure 5-9 shows the transient response of the triangle frame with Node-Layer method. The Node-Layer method shows better correlation than the other model compared to the method of Hashmi *et al.* method. Both the Node-Layer method and Bending-Division method provide good alternatives to predict dynamic non-linear response of structures subjected to different loads.

It is difficult to model branching structures such as T-joint, etc, by using Hashmi *et al.*'s or Witmer's model. For branching into T structure, the amended model of Hashmi *et al.* needs to be restructured and the restructured model is dedicated to a particular problem and situation only. In addition, the process is tedious. Figure 5-10 shows a typical example, which requires the branching capabilities. The amended model of Hashmi *et al.*, Bending-Division method and Node-Layer method were used to solving the problem.

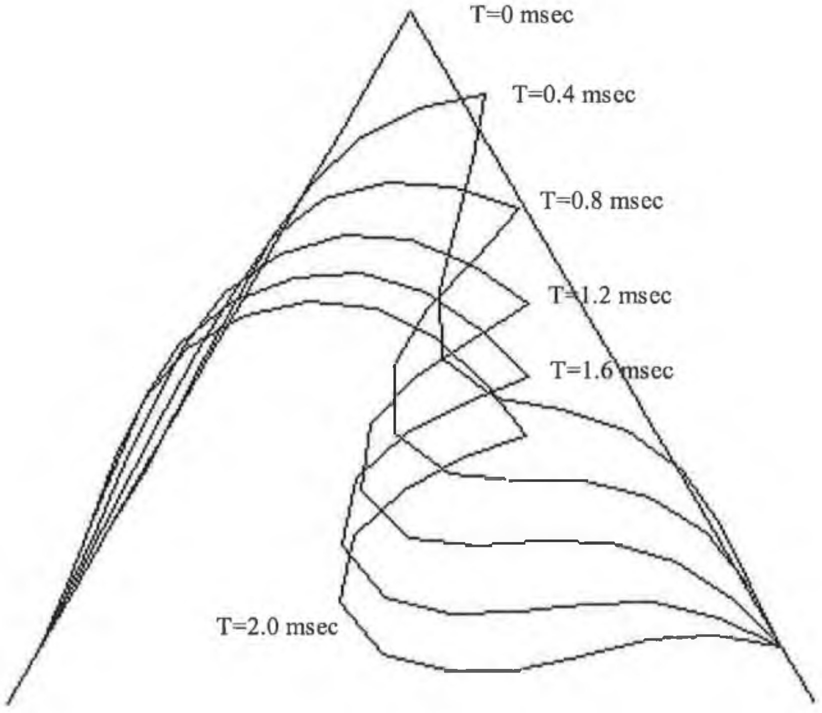


Figure 5-9: Transient response of the triangle frame with Node-Layer method

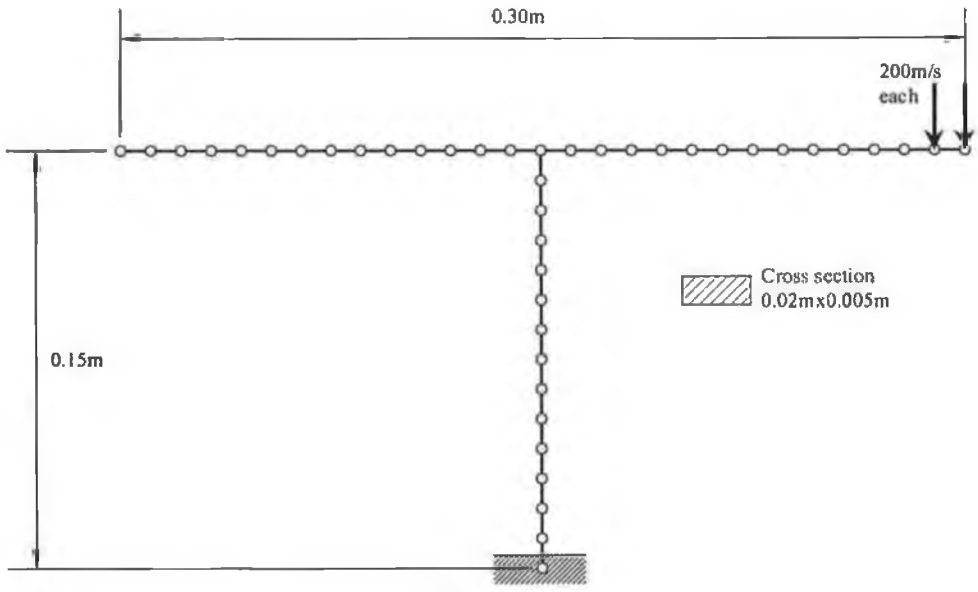


Figure 5-10: T-structure geometry setting and initial loading condition

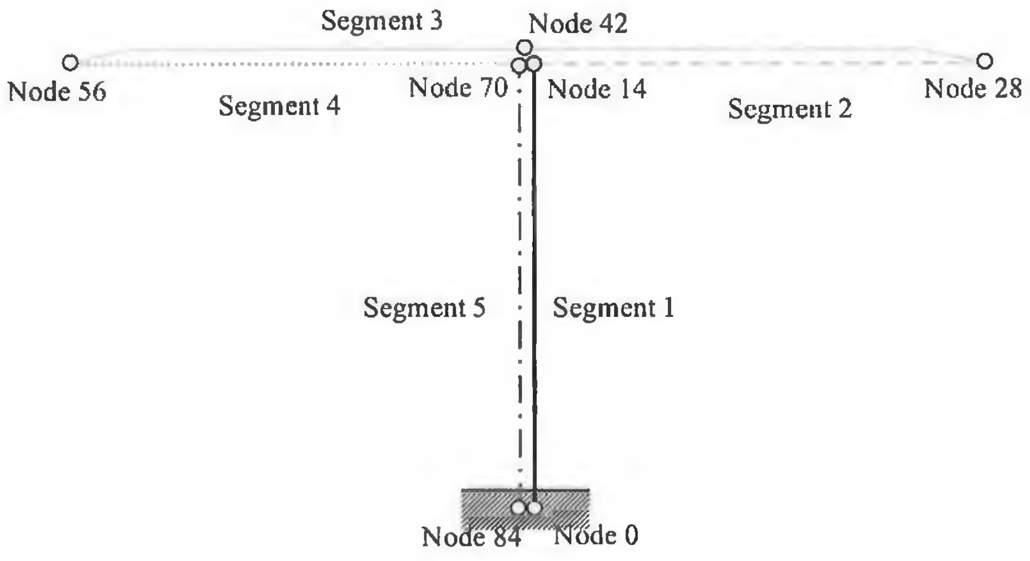


Figure 5-11: FDM model of T-structure problem

Table 5-1: Comparison of present methods with Hashmi *et al.* [1] 's Finite Difference Method. Table values are the deflection of the point of impact with time.

| Time (sec) | Hashmi *et al.*'s FDM | Layer-Node Method | Difference (%) | Bending Division Method | Difference (%) |
|---|---|---|---|---|---|
| 0.000100 | 0.009604 | 0.009558 | -0.472393 | 0.009572 | -0.326370 |
| 0.000200 | 0.014626 | 0.014223 | -2.759755 | 0.014174 | -3.092432 |
| 0.000300 | 0.018935 | 0.018586 | -1.841912 | 0.018560 | -1.975776 |
| 0.000400 | 0.022677 | 0.022221 | -2.010412 | 0.022208 | -2.066372 |
| 0.000500 | 0.025936 | 0.025444 | -1.898023 | 0.025478 | -1.765750 |
| 0.000600 | 0.029039 | 0.028434 | -2.085102 | 0.028491 | -1.889639 |
| 0.000700 | 0.031915 | 0.031377 | -1.684761 | 0.031352 | -1.763046 |
| 0.000800 | 0.034678 | 0.034035 | -1.854140 | 0.033970 | -2.043054 |
| 0.000900 | 0.037184 | 0.036583 | -1.616248 | 0.036510 | -1.811438 |
| 0.001000 | 0.039475 | 0.038956 | -1.316175 | 0.038877 | -1.516799 |
| 0.001100 | 0.041598 | 0.041112 | -1.167621 | 0.040954 | -1.546389 |
| 0.001200 | 0.043577 | 0.043148 | -0.985218 | 0.042809 | -1.763945 |
| 0.001300 | 0.045398 | 0.044962 | -0.959874 | 0.044522 | -1.928785 |
| 0.001400 | 0.047082 | 0.046643 | -0.931737 | 0.046138 | -2.003261 |
| 0.001500 | 0.048639 | 0.048183 | -0.937201 | 0.047627 | -2.080320 |
| 0.001600 | 0.050041 | 0.049623 | -0.835653 | 0.048976 | -2.128300 |
| 0.001700 | 0.051350 | 0.050946 | -0.785812 | 0.050229 | -2.183257 |
| 0.001800 | 0.052537 | 0.052149 | -0.738559 | 0.051371 | -2.219476 |
| 0.001900 | 0.053594 | 0.053197 | -0.741529 | 0.052385 | -2.257229 |
| 0.002000 | 0.054564 | 0.054157 | -0.746377 | 0.053311 | -2.297451 |
| 0.002100 | 0.055467 | 0.055035 | -0.780244 | 0.054161 | -2.354558 |
| 0.002200 | 0.056295 | 0.055831 | -0.823752 | 0.054939 | -2.408039 |
| 0.002300 | 0.057041 | 0.056566 | -0.833221 | 0.055655 | -2.430670 |
| 0.002400 | 0.057736 | 0.057248 | -0.845437 | 0.056360 | -2.383271 |
| 0.002500 | 0.058366 | 0.057877 | -0.838580 | 0.056989 | -2.360294 |
| 0.002600 | 0.058958 | 0.058456 | -0.851438 | 0.057545 | -2.396014 |
| 0.002700 | 0.059527 | 0.058995 | -0.894395 | 0.058030 | -2.516010 |
| 0.002800 | 0.060063 | 0.059514 | -0.914222 | 0.058460 | -2.669450 |
| 0.002900 | 0.060541 | 0.059985 | -0.919225 | 0.058872 | -2.756952 |
| 0.003000 | 0.060931 | 0.060397 | -0.876083 | 0.059242 | -2.772025 |
| 0.003100 | 0.061238 | 0.060730 | -0.828269 | 0.059537 | -2.776814 |
| 0.003200 | 0.061461 | 0.061004 | -0.743026 | 0.059788 | -2.722249 |
| 0.003300 | 0.061656 | 0.061235 | -0.683181 | 0.060009 | -2.671155 |
| 0.003400 | 0.061869 | 0.061456 | -0.668200 | 0.060215 | -2.674020 |
| 0.003500 | 0.062061 | 0.061665 | -0.636768 | 0.060385 | -2.700558 |
| 0.003600 | 0.062220 | 0.061863 | -0.574960 | 0.060526 | -2.723021 |
| 0.003700 | 0.062342 | 0.062029 | -0.502362 | 0.060605 | -2.786068 |
| 0.003800 | 0.062409 | 0.062132 | -0.444189 | 0.060644 | -2.828591 |
| 0.003900 | 0.062456 | 0.062190 | -0.425697 | 0.060668 | -2.862783 |
| 0.004000 | 0.062485 | 0.062237 | -0.397566 | 0.060666 | -2.911270 |
| | | Average | -1.021233 | | -2.284073 |

Table 5-2: Coincident nodes of FDM model of T-structure problem

| Segments (node range) | Coincide node with Node *i* | *i* range |
|---|---|---|
| 1 (0…14) & 5 (70…84) | Node (84-*i*) | *i* = 0…13 |
| 2 (14…28) & 3 (28…56) | Node (56- *i*) | *i* = 15…27 |
| 3 (28…56) & 4 (56…70) | Node (112- *i*) | *i* = 43…55 |

To model the problem with Witmer *et al.*'s method, duplicated links and nodes has to be created. Figure 5-11 shows an exaggerated display of the model. The model consists of 5 segments. Thus, the structure starts from node 0 and ends with node 84. The segments are connected at nodes 14, 28, 56 and 70. Segment 1 and segment 5 are with exact same length at exact same location, while segment 2 and segment 4 lay on segment 3. All segments have cross section of 0.01 m x 0.005 m, which is half of the base length. With this arrangement, node 14, node 42 and node 70 are coincident location nodes, so do node 0 and node 84, and many others. Table 5-2 shows coincident-nodes with different pairs of segments.

All coincident nodes are considered bounded together, thus they have to move with the same displacement, velocity and acceleration together. To achieve this, all individual nodes and links are considered in the same way as those where no bonding between/among coincident nodes. Then, the predicted displacement at next time step of coincident nodes is synchronised by taking the average displacement of all coincident pair of nodes.

To achieve such implementation, user has to amend the execution code of the processor by inserting the above-mentioned condition. This can be achieved without any intervention to the processor coding, but will make the pre-processor and processor development very inefficient and difficult to handle. Furthermore, the modelling methodology will become very complicated.

Figures 5-12, 5-13 and 5-14 are the predicted transient responses respectively. The material chosen is typical aluminium with linear elastic-multi linear strain hardening. Again it is evident that the Node-Layer method have better correlation to the amended model of Hashmi *et al.* than the Bending-Division method. This is as expected because the Bending-Division method is generally an approximate method compared to the Node-Layer method. It idealises that the effect of bending at a node of a link is shared equally by two closest-connected links. The Node-Layer method takes care of all different contributions of bending towards each individual link. For the above problem, the amended model of Hashmi *et al.*, the Bending-Division method and the Node-Layer method used 64 seconds, 50 seconds and 52 seconds processing time respectively. Generally, for complicated structures, the Node-Layer method and the Bending-Division method show better performance in term of processing time compared to the amended model of Hashmi *et al.*. Both the methods overwhelm when solving more complicated branching problems. Minor restructuring of the coding is required with the amended model of Hashmi *et al.* when handling different problems, even changing the number of nodes. Extra attention is needed, as the connection of links must be made with node-I in the link with node-J in another link or vice-versa. Node-I to node-I connection and node-J to node-J connection are prohibited with the model of Hashmi *et al.*

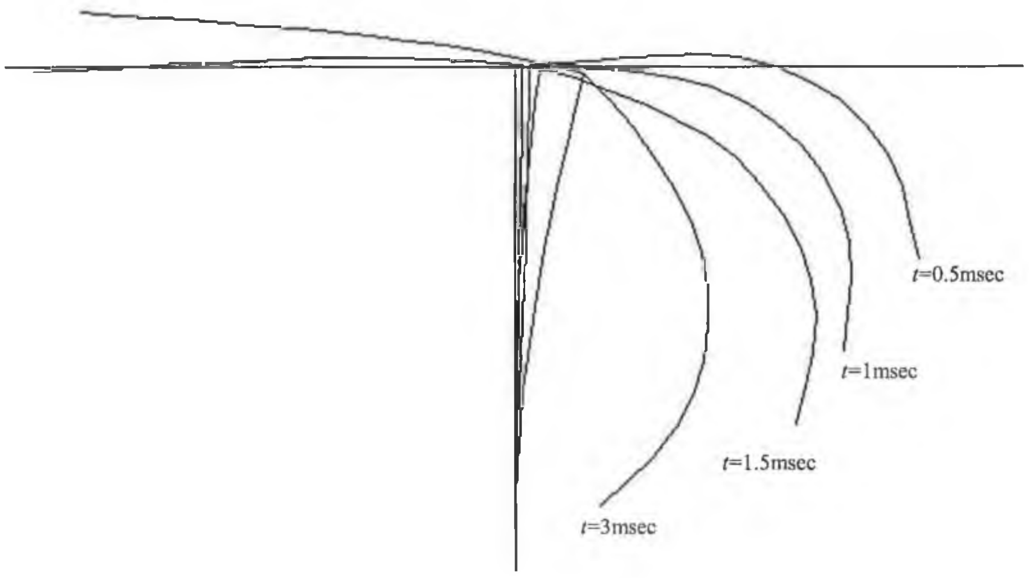


Figure 5-12: Transient response of T-structure with Hashmi *et al.*'s method

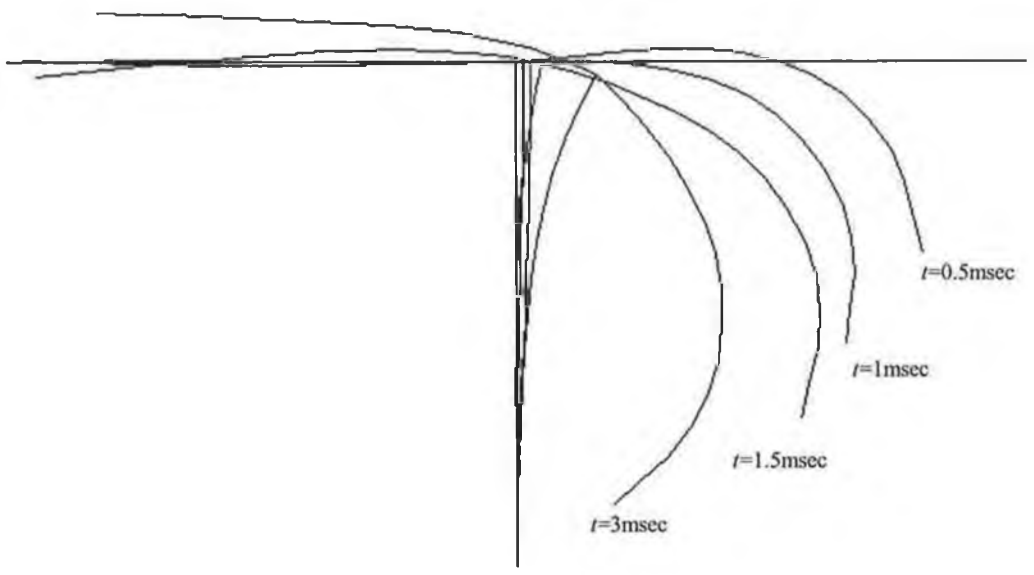


Figure 5-13: Transient response of T-structure with Bending-Division method

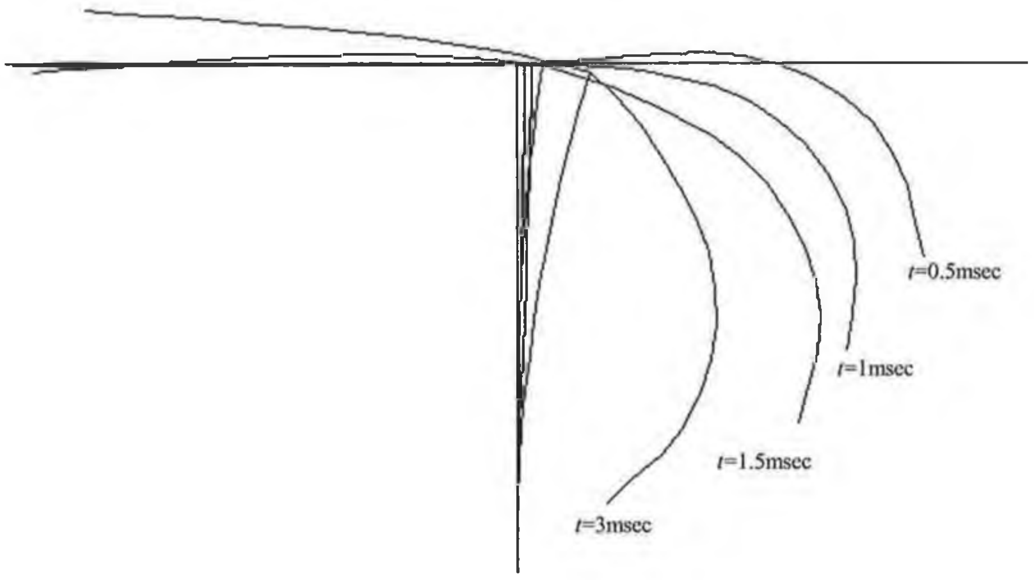


Figure 5-14: Transient response of T-structure with Node-Layer method

Figure 5-15 shows a branching with closing loop problem. The material property is again the idealised linear elastic-multi linear strain hardening aluminium. The ring and the structure have the same cross-section dimension. The ring is having initial velocity of 30m/s and travelling horizontally towards the structure. The contact algorithm involved is beyond the discussion of this section. Figure 5-16 shows the transient dynamic responses of the ring and the structure by using the Node-Layer model. The amended FD model of Witmer *et al.* and Hashmi *et al.* cannot solve this problem.

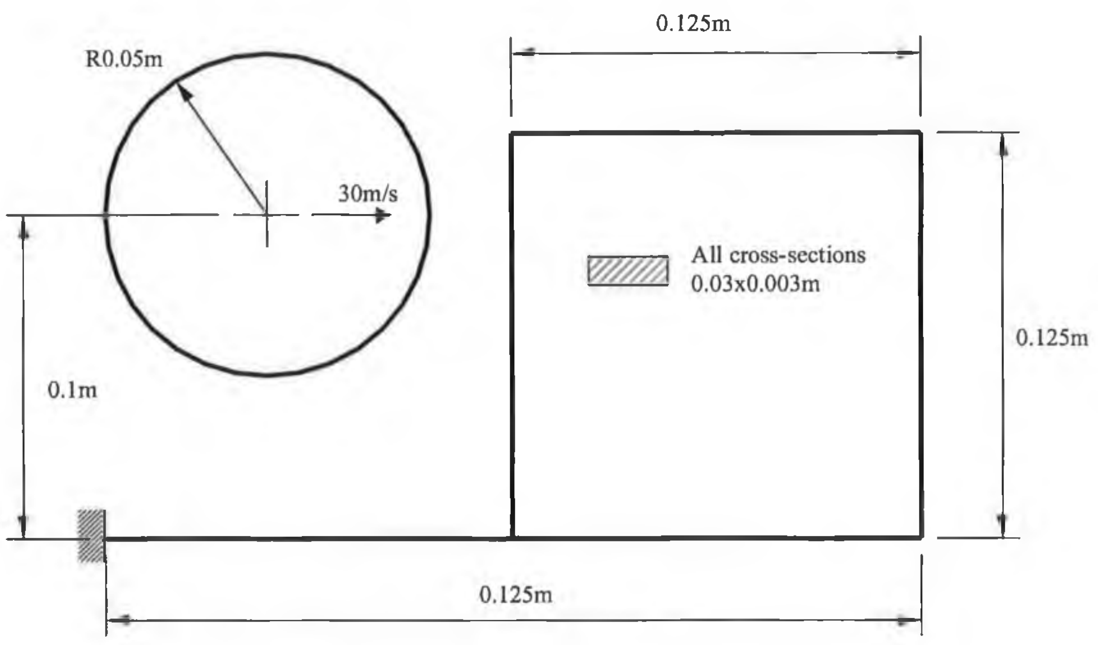


Figure 5-15: Multi objects problem with branching and closed loop structure, geometry setting and initial loading condition

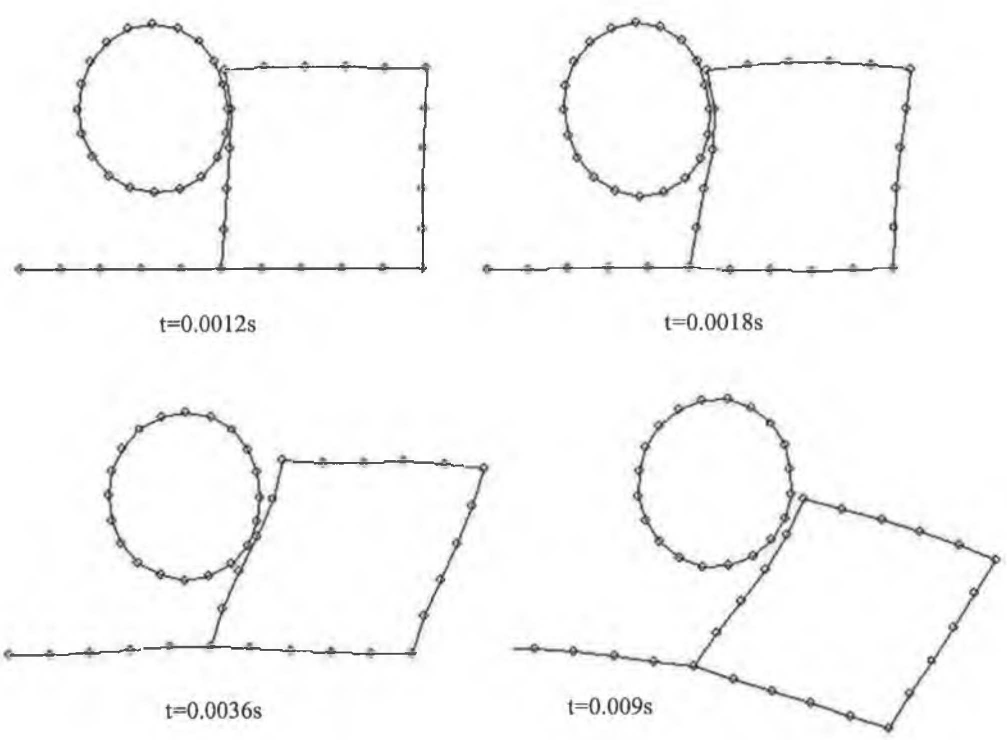


Figure 5-16: Time history responses of the ring and the branching closed loop structure

The numerical example used in section 4.5.11, is a tapered cantilever beam subjected to impact loading, is revisited in this section. The yielded numerical solution will be compared with the experimental results. The model parameters have been amended to suit the experimental set-up [224]. The meshing is refined to 40 nodes with 39 elements. The impulsive load is assumed to have equally distributed at the first two links from the tip of the cantilever. The efficiency of loading is assumed to be 95 percent. With this consideration, the initial velocity of the nearest three nodes to the tip is calculated to be 66.16056 m/s, 65.33864 m/s and 45.34835 m/s respectively. The material stress-strain relation is shown in Figure 4-7, with strain rate sensitivity. The strain rate sensitivity parametric values are 75 as D and 0.1 as P. The numerical result is compared with re-plotted Hashmi's experimental result [224]. The numerical and experimental results are shown in Figure 5-17. The overall profile of the transient response up to 0.016 s is shown in Figure 5-18.

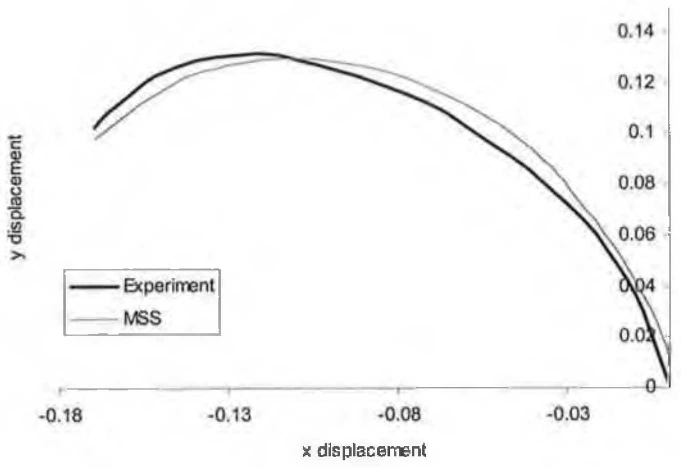


Figure 5-17: Comparison of the experimental and numerical response at t=16 ms

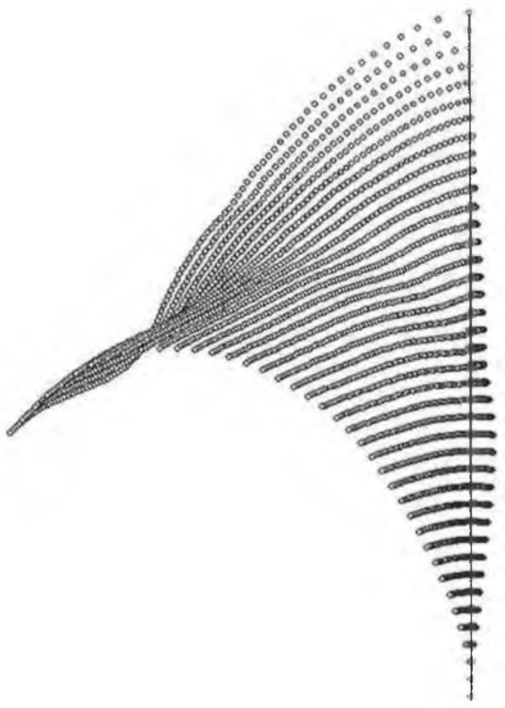


Figure 5-18: Overall motion profile of the tapered cantilever problem

### 5.3.6 DISCUSSIONS AND SUMMARIES

Two general similar, Eulerian-Lagrangian hybrid finite difference hydrocodes are developed for large deformation simulation. They are capable of handling almost all 1-D structure problems in 2-D space. The methods can predict large dynamic elastic and post-elastic responses and permanent deformations of representable structures with 1-D models. Comparison has been made with the finite difference model of Hashmi *et al.* and experimental results. Excellent correlation is exhibited.

The new models are proven to be more efficient in terms of computational modelling efforts while solving complex shapes, including branching to multiple links. The Node-Layer method recorded better performance in terms of accuracy of prediction, while the Bending-Division method better in terms of processing time. Some of the simple branching problems can be solved with the amended model of Hashmi *et al.*, but amendment involved is problem-oriented. Thus, any change in a problem requires amendment of the coding. Both the Node-Layer and Bending-Division method outperformed the amended model of Hashmi *et al.* Both the methods can be extended to cope in a 3-D space with some extra efforts. Similar scheme is applicable on 3-D models.

## 5.4 CENTRAL DIFFERENCE TEMPORAL INTEGRATION

The acceleration is evaluated directly at each time step by using the governing equations. A central difference method is applied to generate the corresponding velocity and displacement vectors for the succeeding time steps. The following relations are used in describing the instantaneous displacement, velocity and acceleration.

$$\mathbf{V}_{t+1/2} = \mathbf{V}_{t-1/2} + \Delta t \mathbf{A}_t \tag{5-28}$$

$$\mathbf{U}_{t+1} = \mathbf{U}_t + \Delta t \mathbf{V}_{t+1/2} \tag{5-29}$$

thus,

$$\mathbf{A}_t = \frac{\delta \mathbf{U}_{t+1} - \delta \mathbf{U}_t}{\Delta t^2} \tag{5-30}$$

The velocity at time $t$=-1/2 time instance is equal to the initial velocity applied ($t$=0). Similar scheme is applied to obtain the change in rotational movement. Figure 5-19 shows the relations in graphical form. Both the FDM and the MSS incorporate the explicit central difference temporal integration scheme.

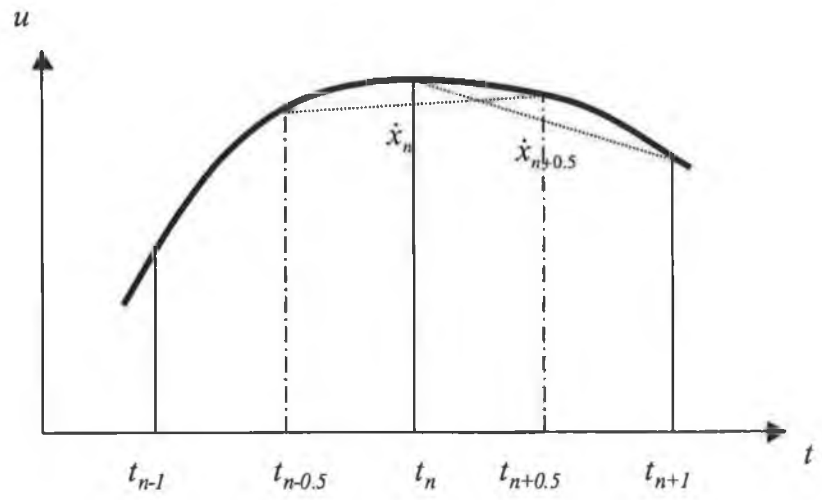


Figure 5-19: Central Difference Temporal Integration Scheme

## 5.5 NUMERICAL STABILITY OF CENTRAL DIFFERENCE TEMPORAL INTEGRATION SCHEME

The Explicit temporal integration scheme such as central difference is conditionally stable. The instability arises from the unrealistic deformation of an element in a single time step, which is totally prohibited in both modelling and real life. Before going further, the stress wave propagation velocity of the critical element need to be known.

### 5.5.1 STRESS WAVE PROPAGATION VELOCITY

The stress wave propagation through a 1-D structural element is to be studied in this section. The element is consisting of two nodes at two ends, namely A and B respectively. An impulse is applied to the end A of the element with constant velocity *v*. After time *t*, the length of the bar compressed will be *ct*, where *c* is the speed with which pulses are transmitted along the element. The compression of the length *ct* is *vt*. Thus, the compressive strain, $\varepsilon$, can be calculated using the following equation.

$$\varepsilon = \frac{vt}{ct} \tag{5-31}$$

Hence, the applied compressive stress, $\sigma$, is

$$\sigma = E\varepsilon = {}^{Ev}\!/_{c} \tag{5-32}$$

where $E$ is the Young's modulus. Let the area of the cross section is denoted as $A$, and the element density as $\rho$. Then the impulse applied at A and the momentum acquired are by equations as,

$$impluse = A\sigma \cdot t \tag{5-33}$$

$$momentum = A\rho ct \cdot v \tag{5-34}$$

Since these two quantities are equal,

$$\sigma = \rho cv \tag{5-35}$$

Thus the stress wave propagation speed is given by

$$c = \sqrt{{}^{E}\!/_{\rho}} \tag{5-36}$$

Typical values of *c* for metals are about 3000 m/s.

### 5.5.2 STABILITY-STRESS WAVE PROPAGATION SPEED RELATION

The temporal integration discretises the time, a continuous quantity in reality. Discretised time interval, $\Delta t$, is a very small fragment of the total interested time frame. $\Delta t$ is usually called the time step. The 1D element described in previous section with A and B as the ends is considered. An impulse is applied at A at $t = 0$ second, which yields a constant velocity, *v*, of movement at the end A. After one time step, node A moves a distance of $v\Delta t$. The response of node B at that time step is predicted based on the compressive stress propagation created by the blow.

For $\Delta t < L/c$, where $L$ is the length of the element, the compressive stress front is still within the element and the response of node B is predicted based on this stress. Whereas, for $\Delta t > L/c$, the stress wave front travels through the length of the element in $\Delta t$ and the stress value at node B cannot be calculated independently, thus creating instability.

The phenomenon can be explained in another way. The node B will start to move after the stress wave front reaches B. Hence, the element experiences the compressive stress of

$$\sigma = \frac{Ev\Delta t}{L} \tag{5-37}$$

up to a maximum stress of

$$\sigma_{\max} = \frac{Ev\,{}^{L}\!/_{c}}{L} = \frac{Ev}{c} \tag{5-38}$$

If $\Delta t > L/c$, then $\sigma > \sigma_{\max}$, which is not valid in reality and will cause instability. Thus, $\Delta t$ must be less than $L/c$ to ensure stability.

Another explanation of the system stability is made based on natural frequency. The time step is said to be bounded by the largest natural frequency of the structure, which in turn is bounded, by the highest frequency of any individual element. The critical time step size is given as

$$\Delta t \le \frac{2}{\omega_{\max}} \tag{5-39}$$

and

$$\Delta t \leq \frac{2}{\omega_{\max}}\left(\sqrt{1+\xi^2}-\xi\right) \tag{5-40}$$

if structural damping is included in term of $\xi$. The $\omega_{\max}$ is the highest natural frequency and $\xi$ is the damping ratio. The ratio is stated as

$$\xi = \frac{C}{C_{cr}} \tag{5-41}$$

where $C$ and $C_{cr}$ are the damping coefficient and the critical damping coefficient of the system.

For the lumped-mass 1D element, the following relations can be found.

$$K = \frac{EA}{L} \tag{5-42}$$

$$m = \frac{\rho AL}{2} \tag{5-43}$$

where $K$ is the element stiffness and $m$ is the mass. The natural frequency of the element is defined with the following equation.

$$\omega = \sqrt{\frac{K}{m}} \tag{5-44}$$

Critical damping of 1D element occurs when

$$\left(\frac{C}{2m}\right)^2 = \frac{K}{m} \tag{5-45}$$

In other words, the critical damping is said to occur when frequency of damped vibration is zero. The distributed mass moves back rapidly to its equilibrium position in the shortest possible time. With the above relations, the critical damping coefficient can be expressed as

$$C_{cr} = 2m\sqrt{\frac{K}{m}} = 2m\varpi \tag{5-46}$$

Equation (5-44) can be expressed in term of stress wave propagation speed, $c$, and the following equation is obtained.

$$\omega = \frac{2c}{L} \qquad (5\text{-}47)$$

Substituting the above relation into Equation (5-39), the maximum stable time step size, $L/c$, is obtained.

### 5.5.3 STABILITY CRITERION

Courant-Friedrichs-Levy stability criterion states that, the time step size must be smaller than $l/c$, where $l$ is the minimum length of all elements, and $c$ is the maximum stress wave propagation speed. Generally, no stability proof is available for time integration of non-linear problems. Thus, a safety factor is used when suggesting stable time step size and 0.9 to 0.7 is normally used. The critical time step size may thus be as follows,

$$\Delta t_{maz} = \frac{0.9l}{c} = 0.9l\sqrt{\frac{\rho}{E}} \qquad (5\text{-}48)$$

## 5.6 STRESS-STRAIN RELATIONS (MATERIAL MODELLING)

An efficient algorithm was developed in this work for handling multi-linear elastic-multi-linear strain hardening material property. Thus, a material can be idealised to have properties ranging from simple elastic-perfectly plastic to non-linear elastic-plastic. Kinematic hardening and hysteresis effects are also included. The strain rate sensitivity model according to Cowper and Symonds [225] is incorporated.

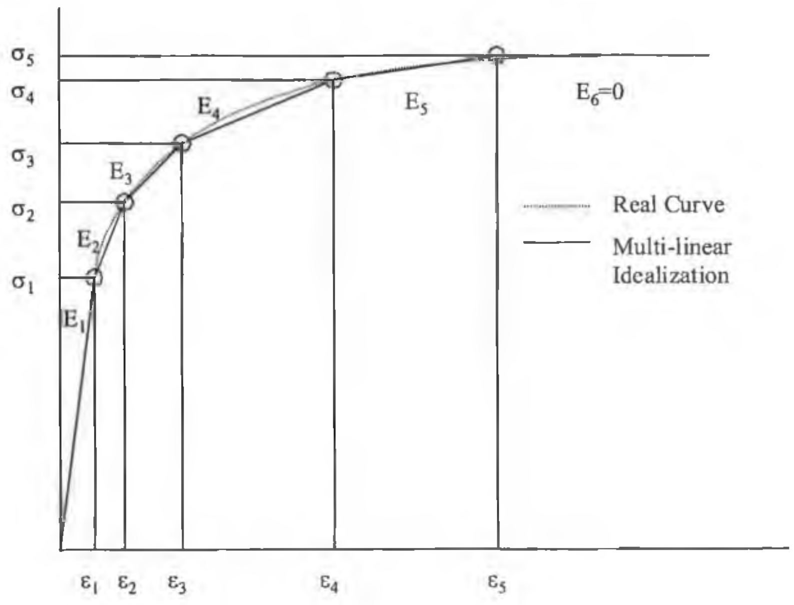


Figure 5-20: Typical stress-strain relationship with multi-linear idealisation

The development of the stress-strain relations is based on the theory of plasticity, which states that part of the body has yielded and part is still elastic when material stress level exceeds the elastic point in the stress-strain curve. Figure 5-20 shows a typical stress-strain relationship and the corresponding multi-linear strain hardening idealisation. Applying the theory of plasticity and assuming that part of the body will yield and having perfect plastic deformation (E=0), whereas the other continuing the same elastic deformation ($E=E_1$), after a material reaches its elastic point ($\sigma_1$ and $\varepsilon_1$ in Figure 5-20). The effects of such arrangement are responsible for the change of E, the slope of the stress-strain curve, after the elastic point. Thus, equations below are developed.

$$\sigma = \sum_{k=1}^{n} \sigma_k R_k \tag{5-49}$$

where

$$\sigma_k = \begin{cases} E_1\varepsilon\,, & |E_1\varepsilon| \le E_1\varepsilon_k \\ E_1\varepsilon_k\,, & E_1\varepsilon > E_1\varepsilon_k \\ -E_1\varepsilon_k\,, & E_1\varepsilon < -E_1\varepsilon_k \end{cases} \tag{5-50}$$

$$R_k = \frac{E_k - E_{k+1}}{E_1} \tag{5-51}$$

These equations are used to obtain the proper stress (σ) with the corresponding strain (ε). The term *k* denotes the *k*th segment of the multi linear idealised stress-strain curve. The parameter *n* is the maximum number of segments, n=5 as in the case of Figure 5-20. The $R_k$ is the fraction of a body, which will yield at $\varepsilon_k$. Linear elastic-multi linear strain hardening is handled with Equations (5-49) to (5-51) but hysteresis and Bauschinger effects are not included. The change of strain, $\delta\varepsilon_p$, at a particular time instance *t*, is considered instead of the strain alone. The fractional stresses (Equation (5-50)) corresponding to the $\delta\varepsilon_t$ is calculated with Equation (5-52).

$$\sigma_{k,t} = \begin{cases} \sigma_{k,t-1} + E_1\delta\varepsilon_t\,, & \left|\sigma_{k,t-1} + E_1\delta\varepsilon_t\right| \le E_1\varepsilon_k \\ E_1\varepsilon_k\,, & \sigma_{k,t-1} + E_1\delta\varepsilon_t > E_1\varepsilon_k \\ -E_1\varepsilon_k\,, & \sigma_{k,t-1} + E_1\delta\varepsilon_t < -E_1\varepsilon_k \end{cases} \tag{5-52}$$

Equation (5-52) is based on the assumption that the deformation in the hysterisis region follows the elastic route and the value of the Young's modulus being unaltered by the plastic deformation [226]. A typical linear elastic-multi linear strain hardening stress-strain diagram is shown in Figure 5-21. The effects, yielded from the stress-strain formulation so far, are shown as well. Equations (5-49) to (5-52) are amended to cope with multi linear elastic – multi linear strain hardening, and yield the following equations.

$$\sigma_{k,t} = \begin{cases} \sigma_{e,t}, & \left|\sigma_{e,t}\right| \le E_1\varepsilon_k, k \le n_e \\ E_1\varepsilon_k, & \sigma_{e,t} > E_1\varepsilon_k, k \le n_e \\ -E_1\varepsilon_k, & \sigma_{e,t} < -E_1\varepsilon_k, k \le n_e \\ \sigma_{k,t-1} + E_1\delta\varepsilon_t, & \left|\sigma_{k,t-1} + E_1\delta\varepsilon_t\right| \le E_1\varepsilon_k, k > n_e \\ E_1\varepsilon_k, & \sigma_{k,t-1} + E_1\delta\varepsilon_t > E_1\varepsilon_t, k > n_e \\ -E_1\varepsilon_k, & \sigma_{k,t-1} + E_1\delta\varepsilon_t < -E_1\varepsilon_t, k > n_e \end{cases} \tag{5-53}$$

$$\sigma_{e,t} = \begin{cases} \sigma_{e,t-1} + E_1\delta\varepsilon_t, & \left|\sigma_{e,t-1} + E_1\delta\varepsilon_t\right| \le E_1\varepsilon_{n_e} \\ E_1\varepsilon_{n_e}, & \sigma_{e,t-1} + E_1\delta\varepsilon_t > E_1\varepsilon_{n_e} \\ -E_1\varepsilon_{n_e}, & \sigma_{e,t-1} + E_1\delta\varepsilon_t < -E_1\varepsilon_{n_e} \end{cases} \tag{5-54}$$

where $n_e$ is the segment number, which ends the non-linear elastic deformation.

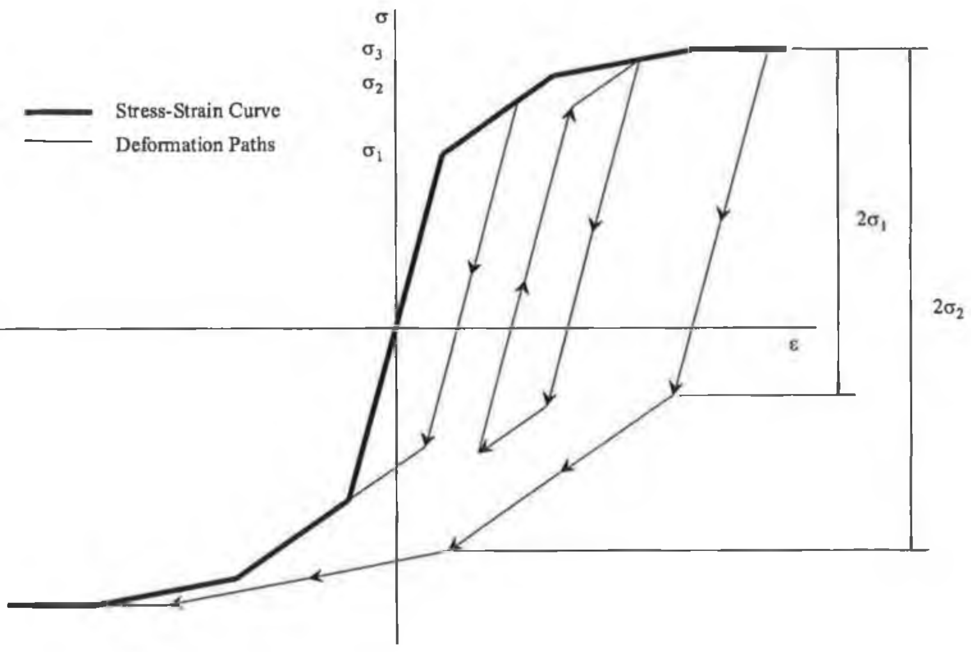


Figure 5-21: Typical elastic-multi linear strain hardening stress-strain relation and typical loading-unloading responses

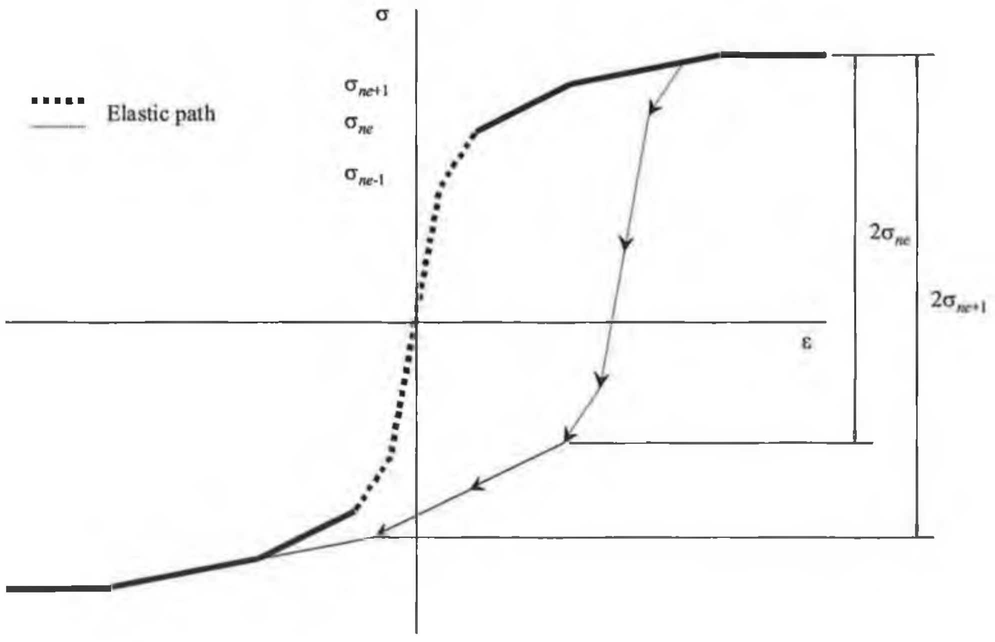


Figure 5-22: Typical multi linear elastic-multi linear strain hardening stress-strain relation and typical loading-unloading response.

Figure 5-22 shows an exaggerated multi linear elastic – multi linear strain hardening stress-strain diagram. A typical unloading after the elastic point with further compression is shown.

To incorporate strain rate sensitivity, Cowper and Symonds model [225] is used. The yield stress is scaled with the following equation.

$$\sigma'_k = \left[1 + \left(\frac{\dot{\varepsilon}}{D}\right)^{1/P}\right]\sigma_k \qquad (5\text{-}55)$$

The new yield stress for different segments is verified with Equation (5-55) each time before applying Equations (5-49) to (5-54). The parameters $D$ and $P$ have to be determined by experiments.

## 5.7 BOUNDARY CONDITIONS

The free end of a structure can have three types of end conditions, namely freed, fixed and pivoted. The boundary conditions handling in the FDM is different from that in the MSS. This is due to the nature of the incorporated elements. Because of implementation nature of the FDM as well as Witmer *et al.*'s and Hashmi et al's method, a virtual element is required to handle the end conditions. The method requires the model to start with the first node of the first element and end with the last node of the last element. This again shows the inefficiency of the method. On the other hand, the MSS does not require such an approach.

### A. Free End

A free end is the simplest boundary condition. No constraint is applied at the edge node of the end element. Which means, no axial force in the element, no shear force and no bending moment at the edge node. The FDM implementation for the first and last elements will be like the following:

```
node[0].dDeltTheta=0;
node[1].dDeltTheta=0;
link[1].Q=0;
link[1].dDelts=0;
```

Coding 5-1: Boundary conditions of first element with the FDM for free end

```
node[iNoOfNode+1].dDeltTheta=0;
node[iNoOfNode].dDeltTheta=0;
link[iNoOfNode+1].Q=0;
link[iNoOfNode+1].dDelts=0;
```

Coding 5-2: Boundary conditions of last element with the FDM for free end

node[0] and link[iNoOfNode+1] are virtual node and link, which do not exist in the model. From the implementation, the change of the angular difference between the end link and the virtual link, *dDeltTheta*, is always zero. The shear force, *Q*, and the change of link length, *dDelts*, are zero as well.

For the MSS implementation, only the bending moment of the end node is required to set to zero. The shear force and the change of element length are calculated as zero.

## B. Fixed End

A fixed end is defined when the structure's end is "planted" into a rigid solid boundary. The end is constrained to all degree of freedom. Figure 5-23 shows such a configuration.

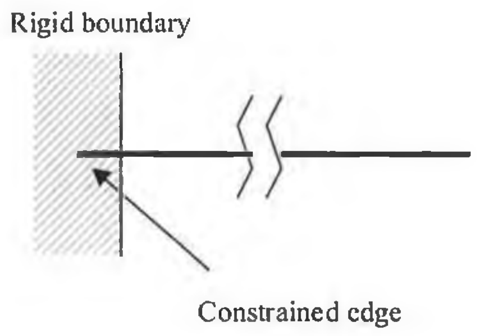


Figure 5-23: Fixed end

Figure 5-24 shows the FDM treatment of the fixed end. The fixed-end element is planted into the rigid boundary.

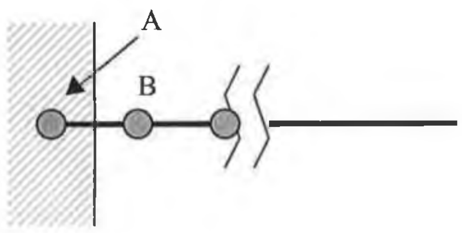


Figure 5-24: FDM fixed end idealisation

The boundary conditions are applied to ensure that the node B is restrained to move except in the axial direction. The axial movement has double effects, which is the mirror effect from the rigid boundary. The mirror effect is only valid with half-link configuration. The following is the implementation code in FDM of handling fixed ends at the first and the last elements respectively.

```
link[1].dDelts=2*(node[1].dx*link[1].cosTheta+node[1].dy*link[1].sinTheta);
Vr=node[1].dx*link[1].sinTheta-node[1].dy*link[1].cosTheta;
node[1].dx=node[1].dx-Vr*link[1].sinTheta;
node[1].dy=node[1].dy+Vr*link[1].cosTheta;
node[1].dDeltTheta=0;
link[1].Q=0;
```

Coding 5-3: FDM first element fixed end boundary conditions

```
link[iNoOfNode+1].dDelts=-2*(node[iNoOfNode].dx*
   (link[iNoOfNode+1].cosTheta)+
    node[iNoOfNode].dy*link[iNoOfNode+1].sinTheta);
Vr=node[iNoOfNode].dx*link[iNoOfNode+1].sinTheta-
    node[iNoOfNode].dy*(link[iNoOfNode+1].cosTheta);
node[iNoOfNode].dx=node[iNoOfNode].dx-Vr*link[iNoOfNode+1].sinTheta;
node[iNoOfNode].dy=node[iNoOfNode].dy+Vr*link[iNoOfNode+1].cosTheta;
node[iNoOfNode].dDeltTheta=0; // no use ... only important for node[n+1]
link[iNoOfNode+1].Q=0;
```

Coding 5-4: FDM last element fixed end boundary conditions

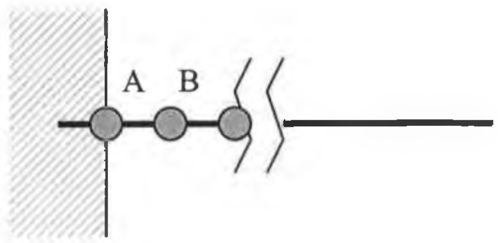


Figure 5-25: MSS fixed end idealisation

A different approach is incorporated in the MSS, which is shown as Figure 5-25. The constrained part is ignored. Node A sits on the surface and is presumed to have zero displacement in all direction, while node B is only free to move in axial direction.

### C. Pivoted End

A pivoted end is simple to achieve by providing all the boundary conditions of a free end except applying constraints of linear motion at the end node in all directions.

## 5.8 INSTANTANEOUS NUMERICAL VALUE VERSUS EFFECTIVE VALUE

The instantaneous numerical values are all quantities at a particular time instant. Of interest are the displacement, velocity, acceleration, axial force, shear force, and bending moment. The numerical solution tends to "stabilise" the displacement response, which resulted from the instantaneous acceleration and velocity. Thus, all but instantaneous velocity and acceleration can be treated as effective value. Vibratory and cyclical types of responses are expected to be formed in velocity and particularly in acceleration profiles. A statistical method in finding the effective velocity and acceleration at a particular time step, based on the displacement profile is suggested.

The effective velocity at a time instance, *t*, is calculate with the following equation,

$$v^t = \frac{\displaystyle\sum_{i=t-\text{int}(n/2)}^{t+\text{int}(n/2)} u^i t^i - \frac{\left(\displaystyle\sum_{i=t-\text{int}(n/2)}^{t+\text{int}(n/2)} u^i\right)\left(\displaystyle\sum_{i=t-\text{int}(n/2)}^{t+\text{int}(n/2)} t^i\right)}{n}}{\displaystyle\sum_{i=t-\text{int}(n/2)}^{t+\text{int}(n/2)} \left(t^i\right)^2 - \frac{\left(\displaystyle\sum_{i=t-\text{int}(n/2)}^{t+\text{int}(n/2)} t^i\right)^2}{n}} \tag{5-56}$$

The value *n* is the number of time steps involved in calculating the $v^t$. The term u$^i$ and $t^i$ are the nodal displacement and respective time at time instant *i*. the int(*a*) is a function which will round the value of *a* to an integer. The development of the above equation is based on Least Squares Method (LSM). The *n* is the parameter mentioned in the MSS Post-Processor in Chapter 4, which is set to 7 as default. The value of *n* should not be too large or too small. The former case will lead to an average velocity of whole processed time frame (*n* = total number of time steps). The latter situation will lead to instantaneous velocity (*n* = 1). Validation is carried out to make sure that the value of *n* is not too large, which is determined by the following equation,

$$n' = \begin{cases} \text{int}(0.15\times iTSI), & \text{if } n > \text{int}(0.15\times iTSI) > 2 \\ 2, & \text{if } \text{int}(0.15\times iTSI) < n < 2 \\ n, & \text{otherwise} \end{cases} \tag{5-57}$$

where $iTSI$ is the number of time step interval. In the same way, the effective acceleration can be found from the effective velocity using Equation (5-58).

$$a^t = \frac{\displaystyle\sum_{i=t-\text{int}(n/2)}^{t+\text{int}(n/2)} v^i t^i - \frac{\left(\displaystyle\sum_{i=t-\text{int}(n/2)}^{t+\text{int}(n/2)} v^i\right)\left(\displaystyle\sum_{i=t-\text{int}(n/2)}^{t+\text{int}(n/2)} t^i\right)}{n}}{\displaystyle\sum_{i=t-\text{int}(n/2)}^{t+\text{int}(n/2)} \left(t^i\right)^2 - \frac{\left(\displaystyle\sum_{i=t-\text{int}(n/2)}^{t+\text{int}(n/2)} t^i\right)^2}{n}} \tag{5-58}$$

where $v^i$ is the effective velocity at time step *i* from Equation (5-18).

The numerical solution in Section 4.5.11, for a tapered cantilever problem, is used to plot the above charts. Both the instantaneous and effective values of both the displacement and acceleration of node 39, the last node at the tip of the cantilever, are plotted. Figure 5-26 shows the instantaneous and effective velocity of the x-component. The instantaneous velocity shows some disturbance, but still within acceptable range, but not for the instantaneous acceleration shown in Figure 5-27. More distorted instantaneous acceleration profile is found in the chart. The instantaneous acceleration goes up to 1.5x10$^6$ m/s$^2$, and down to -1.0x10$^6$ m/s$^2$ at some instance, which is totally unrealistic compared to the effective acceleration, which is in the range of 20000 m/s$^2$ to –1500 m/s$^2$.

As a whole, the LSM effective velocity and acceleration should be used to indicate the real acceleration rather than the instantaneous corresponding values at any time.

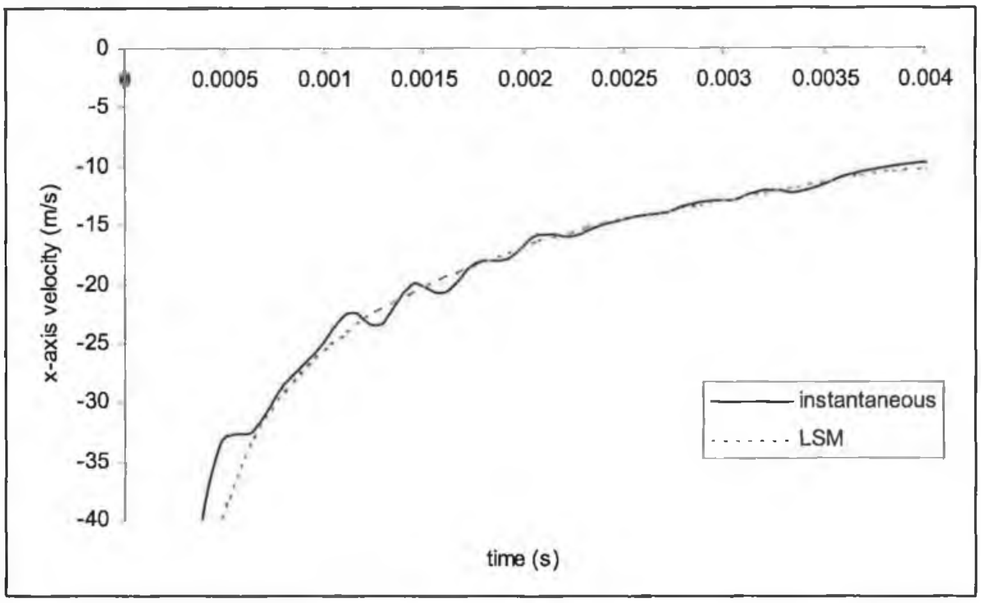


Figure 5-26: Instantaneous and effective velocity of Node 39 of tapered cantilever problem

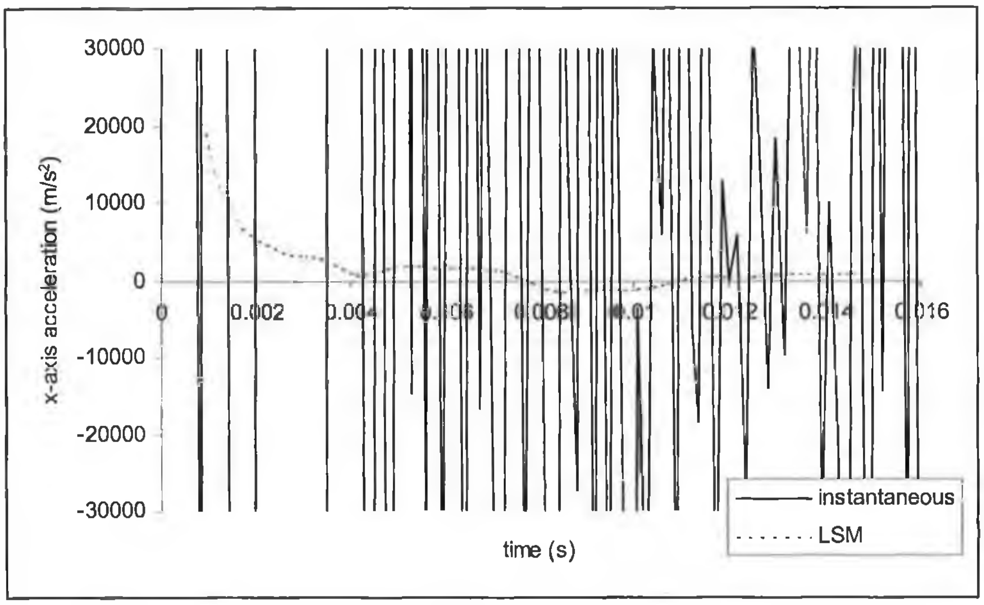


Figure 5-27: Instantaneous and effective acceleration of Node 39 of tapered cantilever problem

## 5.9 FDM CONTACT-IMPACT STRATEGY

A simple contact-impact treatment has been developed in this work. The formulation involves contact-impact of a fixed position rigid bar with the structure. A searching mechanism is implemented to reduce the searching time. The search considers all the elements towards the contact possibility with the rigid bar. The cross-section of the rigid bar is represented with a 2D circle.

### A. Preliminary Contact Search

A preliminary contact criterion is applied to eliminate unrealistic contact situation. The contact algorithm is only activated if the following condition is fulfilled.

$$\sqrt{(x_p - x_i)^2 + (y_p - y_i)^2} \leq R \qquad (5\text{-}59)$$

where $x_p$ and $y_p$ are the location of the centre of the obstacle. The $x_i$ and $y_i$ are the *i*th node location. $R$ is the radius of the effective circular zone, which is defined by the following equation.

$$R = \sqrt{r^2 + l_{\max}^2} \tag{5-60}$$

where $r$ is the obstacle radius and $l_{\max}^2$ is the maximum link length of all elements in the model. If and only if this condition is valid, further procedure is carried out to validate the contact.

## B. Contact Validation

Further validation is needed to check for contact occurrence. Firstly, the normal distance from a link element to the centre of the obstacle has to be calculated. Before that, the perpendicular point on the link element to the contact obstacle centre has to be found with the following equation.

$$x_t = \frac{\frac{x_p}{M} + Mx_i + y_p - y_i}{\frac{1}{M} + M} \tag{5-61}$$

$$y_t = M(x_t - x_i) + y_i \tag{5-62}$$

where $(x_i,\ y_i)$, $(x_t,\ y_t)$, and $(x_p,\ y_p)$ are the co-ordinates of the node *i*, the contact location, and the centre of the obstacle respectively. They are shown in Figure 5-28. *M* is the slope of the link, which is defined as below:

$$M = \frac{y_{i-1} - y_i}{x_{i-1} - x_i} \tag{5-63}$$

Special care has to be taken in dealing with $M = 0$ and $M \approx \propto$. The normal distance, *d*, is calculated with the following equation.

$$d = \sqrt{(x_t - x_p)^2 + (y_t - y_p)^2} \tag{5-64}$$

If contact occurs, all the conditions below have to be fulfilled.

$$d < r \tag{5-65}$$

$$\min(x_i, x_{i-1}) \le x_t \le \max(x_i, x_{i-1}) \tag{5-66}$$

and

$$\min(y_i, y_{i-1}) \le y_t \le \max(y_i, y_{i-1}) \tag{5-67}$$

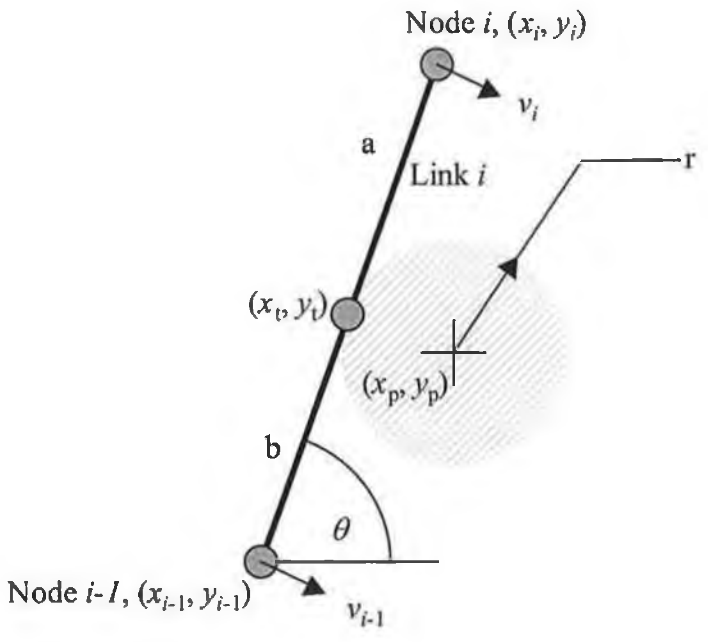


Figure 5-28: FDM contact model

## C. Contact Response

Figure 5-28 is used to assist in predicting the contact response. With the co-ordinate of $x_t$ and $y_t$, the link segment length of $a$ and $b$ can be found easily. Conservation of rotary momentum is used and yields the relationship,

$$\omega = \frac{-m_i v_i a + m_{i-1} v_{i-1} b}{m_i a^2 + m_{i-1} b^2} \tag{5-68}$$

where $\omega$ is the rotary velocity around $(x_i, y_i)$ after the contact and $m_i$ is the mass of node *i*. The velocities of the nodes *i* and *i*-1 are updated respectively with the following equations.

$$v_i = -\omega a \tag{5-69}$$

$$v_{i-1} = \omega b \tag{5-70}$$

## 5.10 MSS CONTACT-IMPACT ALGORITHM

A complete general contact-impact algorithm consists of the following main functions:

- Contact-impact modelling
- Contact searching
- Normal contact-impact response
- Tangential frictional response

The treatment of contact interfaces involving beam elements is most difficult [105]. Contacts on beams can only occur on the physical boundary of the beam not on the reference line. A scheme is proposed to handle such idealised situation. The scheme is capable of handling general contact-impact problem as well. The 2-D contact-impact model is described including the node-node and node-surface contacts. The model can be upgraded to handle 3-D contact problem easily. Section 5.10.2 shows the contact-searching algorithm. Potential voids and overlaps prevention algorithm developed by the authors is illustrated. The normal contact response with kinematics contact algorithm (KCA) developed by the author is presented in section 5.10.3. Besides that, a novel adaptive method designed and developed in this study, namely Adaptive Penalty Method (AP), is included in this section. It has the similar pattern of bouncing back as the well-known penalty method, but avoids the user-defined contact stiffness. The conventional penalty method is presented as well.

According to Zhong and Mackerle [105], the implementation of the classical friction law appears to be more difficult than that of the non-classical friction law introducing displacement dependence. Tangential frictional response of rigid Coulomb's friction model and the general integration of the Contact-Impact Elements are discussed. Energy conservation method is described as well in section 5.10.6. It is used to handle multiple contact nodes contacting an element surface and also a node contacting multiple element surfaces. Numerical examples are shown and comparisons are made among different contact-impact models and the exact solutions. Results with different contact parameters of kinematic contact-impact algorithm are shown.

### 5.10.1 CONTACT-IMPACT MODEL

The contact-impact model is similar to the ANSYS-Contact48/49 element. The contact elements are defined before sending to the processor. The processor scans through all the contact elements and penetration validations are then made. Figure 5-29 shows the nodal orientation of a contact element. The element consists of 3 nodes for a 2-D contact element. The target element nodes could be increased to 3 or more defining a 3-D surface. For handling beam-typed and one-dimensional elements, the thickness is required for the contact algorithm. Figure 5-30 is considered instead, which shows the construction of a contact element to handle uniform dimension contact structure and target structure. The parameters $t_{c,i}$, $t_{c,o}$, $t_{t,i}$ and $t_{t,o}$ are the inner thickness of the contact node K, outer thickness of the contact node K, inner thickness of the target element and outer thickness of the target element respectively. For a more complicated situation such as varying structure dimension, non-symmetrical cross section, etc., the corresponding thickness consideration should be manipulated accordingly. The effective thickness of the element is calculated with the following equation.

$$\begin{aligned} t_{\mathrm{eff,i}} &= t_{\mathrm{c,i}} + t_{\mathrm{t,i}} \\ t_{\mathrm{eff,o}} &= t_{\mathrm{c,o}} + t_{\mathrm{t,o}} \end{aligned} \tag{5-71}$$

The arrangement of the idealised boundaries will avoid any void and overlap at a contact intersection, which will be detailed in the following sub-section. The latter model will be used as reference in the following sub-sections. The former model can be achieved easily by offsetting the target element through the effective thickness when calculating the penetration velocity, etc.

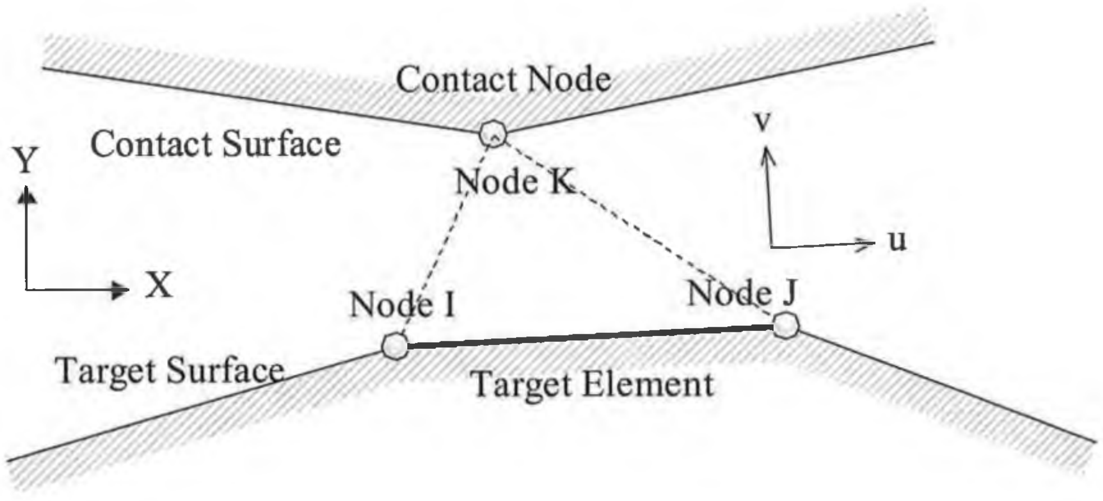


Figure 5-29: Contact element without element thickness

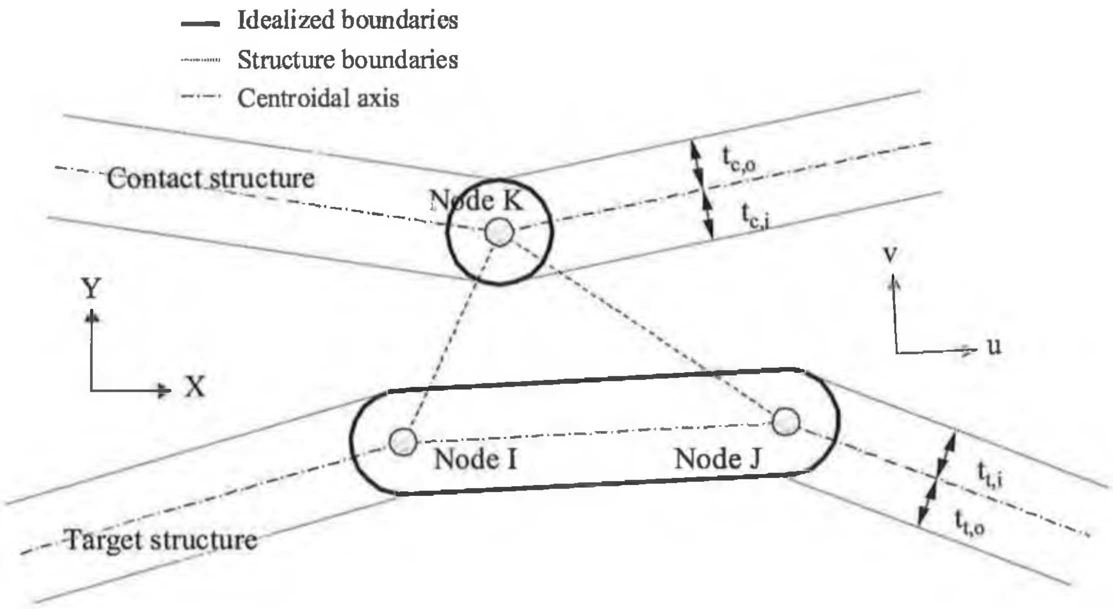
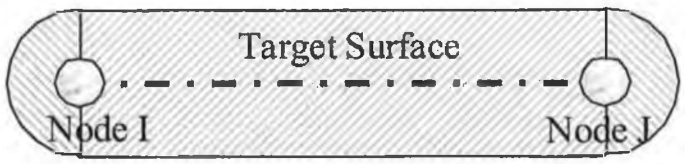


Figure 5-30: Contact element with element thickness

## 5.10.2 CONTACT SEARCHING

The contact searching is a simple systematical consideration of all the contact elements. Contact occurs whenever the contact node penetrates the target surface. If penetration is encountered, normal and tangential contact-impact responses are calculated. Two types of contact are considered, which are node-node contact and node-surface contact.



▨ Node-to-node contact zone

▨ Node-to-surface contact zone

Figure 5-31: Different contact zones

An independent target element of a contact element is defined, when the target element doesn't have any neighbour contact element. The independent target element consists of a node-surface contact zone and two node-node contact zones. Figure 5-31 shows the respective contact zones for a single element contact. Node-surface contact occurs when either condition (5-72) or (5-73) and condition (5-74) are satisfied.

$$0 \le (\mathbf{u}_K - \mathbf{u}_I)^T \hat{\mathbf{n}} < t_{\text{eff,i}}\,, \quad (\mathbf{v}_{Kr})^T \hat{\mathbf{n}} < 0 \tag{5-72}$$

$$0 \ge (\mathbf{u}_K - \mathbf{u}_I)^T \hat{\mathbf{n}} > -t_{\text{eff,o}}\,, \quad (\mathbf{v}_{Kr})^T \hat{\mathbf{n}} > 0 \tag{5-73}$$

$$0 < (\mathbf{u}_K - \mathbf{u}_I)^T \hat{\mathbf{s}} < (\mathbf{u}_J - \mathbf{u}_I)^T \hat{\mathbf{s}} \tag{5-74}$$

where $\mathbf{u}_I$, $\mathbf{u}_J$ and $\mathbf{u}_K$ are the global displacement of node I, J and K respectively. Vector $\hat{\mathbf{n}}$ and $\hat{\mathbf{s}}$ are the unit vectors of local co-ordinate system corresponding to the global co-ordinate system. The $\mathbf{v}_{Kr}$ is the relative velocity of contact node K to the contact surface of the target element. It is defined by the following equation.

$$\mathbf{v}_{\mathrm{Kr}} = \mathbf{v}_{\mathrm{K}} - \mathbf{v}_{\mathrm{I}} - (\mathbf{v}_{\mathrm{J}} - \mathbf{v}_{\mathrm{I}})\left(\frac{(\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I}})^{\mathrm{T}}\hat{\mathbf{s}}}{(\mathbf{u}_{\mathrm{J}} - \mathbf{u}_{\mathrm{I}})^{\mathrm{T}}\hat{\mathbf{s}}}\right) \tag{5-75}$$

Node-node contact can occur at the edges of a target element. Node-node contact can be illustrated as contact-impact of two spheres. The node K and node I are in contact when condition (5-76) is met and condition (5-74) is not valid.

$$\left|\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I}}\right| < t_{\mathrm{eff,i}} + (t_{\mathrm{eff,o}} - t_{\mathrm{eff,i}})\left(\frac{\tan^{-1}\left(\dfrac{-(\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I}})^{\mathrm{T}}\hat{\mathbf{s}}}{(\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I}})^{\mathrm{T}}\hat{\mathbf{n}}}\right)}{\pi}\right) \tag{5-76}$$

For node K and node J contact, condition (5-77) is required instead of (5-76).

$$\left|\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{J}}\right| < t_{\mathrm{eff,i}} + (t_{\mathrm{eff,o}} - t_{\mathrm{eff,i}})\left(\frac{\tan^{-1}\left(\dfrac{(\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{J}})^{\mathrm{T}}\hat{\mathbf{s}}}{(\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{J}})^{\mathrm{T}}\hat{\mathbf{n}}}\right)}{\pi}\right) \tag{5-77}$$

When a contact element is connected with neighbour contact elements, potential voids and overlaps may occur at the intersection of the contact elements. Figure 5-32 shows the potential voids and overlaps in the intersection of two target elements based on element normal projection. The element normal projection method is used in contact type 13 of LS-DYNA3D [102]. Figure 5-33 shows the potential overlaps with the present contact searching procedure and contact-impact element structure described in this section. To handle potential voids and overlaps, ANSYS uses pseudo element algorithm whereas LS-DYNA3D uses nodal point normal projection. To tackle such condition, the relation at the intersection as in Figure 5-34 has been designed in this work. New considerations are included while validating the intersection region of the contact elements. In the following part of this section, attention is given to the development of contact condition at the boundary of contact element A, with element B as its neighbour contact element.

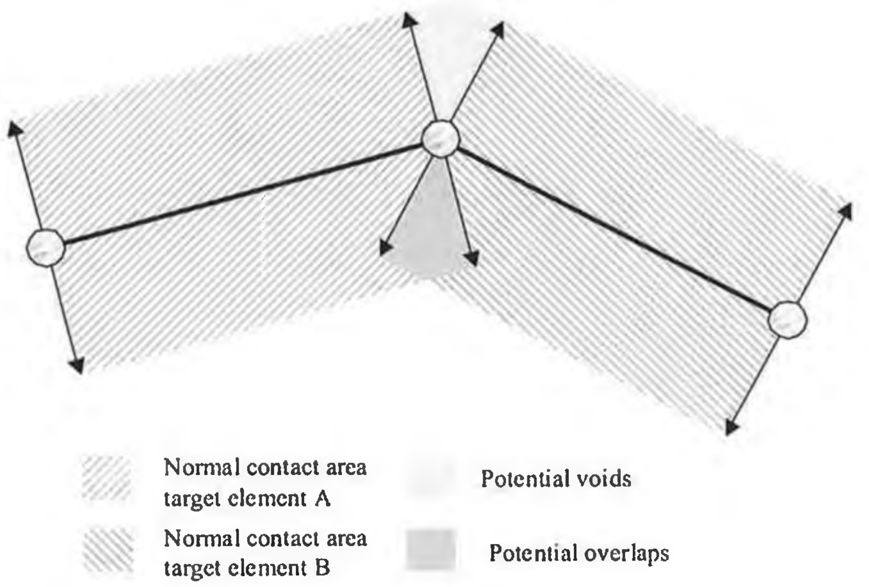

Normal contact area target element A
Normal contact area target element B
Potential voids
Potential overlaps

Figure 5-32: Potential voids and overlaps at the intersection with normal projection method

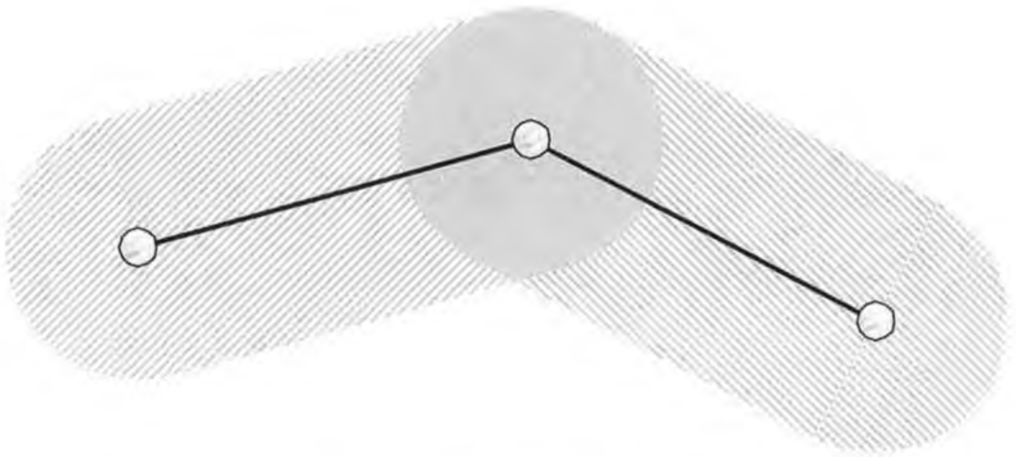


Figure 5-33: Potential overlaps at the intersection with present method

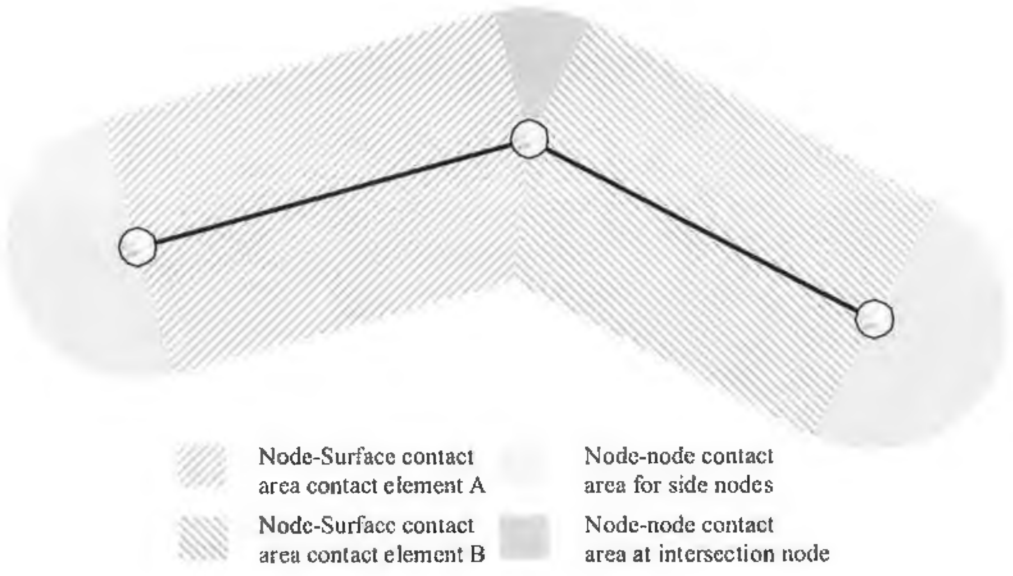

Node-Surface contact area contact element A
Node-node contact area for side nodes
Node-Surface contact area contact element B
Node-node contact area at intersection node

Figure 5-34: Different contact zones after present procurement to prevent voids and overlaps

Initially, the unit contact boundary vector of the two intersected-elements is defined as the following equation.

$$\hat{\mathbf{n}}_{AB} = \frac{\hat{\mathbf{n}}_A + \hat{\mathbf{n}}_B}{\left|\hat{\mathbf{n}}_A + \hat{\mathbf{n}}_B\right|} \tag{5-78}$$

where $\hat{\mathbf{n}}_{AB}$ is the unit contact boundary vector of contact elements A and B. $\hat{\mathbf{n}}_A$ and $\hat{\mathbf{n}}_B$ are the unit normal vector of surface target elements A and B, which are defined in the local co-ordinate system of respective elements. Equation (5-78) is valid if and only if node I of element A is connected to node J of element B or node J of element A is connected to node I of element B. If the condition mentioned above fails, Equation (5-78) has to be changed to Equation (5-79).

$$\hat{\mathbf{n}}_{AB} = \frac{\hat{\mathbf{n}}_A - \hat{\mathbf{n}}_B}{\left|\hat{\mathbf{n}}_A - \hat{\mathbf{n}}_B\right|} \tag{5-79}$$

New condition is needed in addition to Equations (5-72), (5-73), and (5-74) to cope with the potential overlap region shown in Figure 5-33. Equations (5-76) and (5-77) have to be changed to handle any potential void. Condition (5-80) is applied when target element B is connected to target element A at node I. Whereas, condition (5-81) is for target element B connected to target element A at node J.

$$\left(\hat{\mathbf{n}}_{AB}{}^{T}\hat{\mathbf{s}}_{A}\right)\left(\mathbf{u}_{K}{}^{T}\hat{\mathbf{n}}_{A}\right)-\left(\hat{\mathbf{n}}_{AB}{}^{T}\hat{\mathbf{n}}_{A}\right)\left(\mathbf{u}_{K}{}^{T}\hat{\mathbf{s}}_{A}\right)<0,\quad \text{at node I} \tag{5-80}$$

$$\left(\hat{\mathbf{n}}_{AB}{}^{T}\hat{\mathbf{s}}_{A}\right)\left(\mathbf{u}_{K}{}^{T}\hat{\mathbf{n}}_{A}\right)-\left(\hat{\mathbf{n}}_{AB}{}^{T}\hat{\mathbf{n}}_{A}\right)\left(\mathbf{u}_{K}{}^{T}\hat{\mathbf{s}}_{A}\right)>0,\quad \text{at node J} \tag{5-81}$$

For node-surface contact conditions, either condition (5-72) or (5-73), conditions (5-74) and (5-80) and (5-81) are applied. For node-node contact to occur at the intersection, conditions (5-76) and (5-80) or conditions (5-77) and (5-81) are true. In both cases, condition (5-74) has to be invalid. Table 5-3 shows the summary condition requirements for different cases.

Table 5-3: Contact-impact conditions summary

| Contact Element Type | Contact Type | Conditions |
|---|---|---|
| Independent Target Element | Node-node | (5-76) OR (5-78) is TRUE AND (5-74) is FALSE |
| | Node-surface | (5-72) OR (5-73) AND (5-74) are TRUE |
| Dependent Target Element | Node-node | [(5-76) AND (5-80)] OR [(5-77) AND (5-81)] are TRUE AND (5-74) is FALSE |
| | Node-surface | (5-72) OR (5-73) AND (5-74) AND (5-80) AND (5-81) are TRUE |

## 5.10.3 NORMAL CONTACT-IMPACT RESPONSE

Contact-impact contact methods are described in this section. They are the well known conventional Penalty Method, Adaptive Penalty method, Kinematic Contact-impact Method, and Combined Kinematic – Adaptive Penalty Method, which are developed in this work..

### A. Penalty Methods

For the following sections of this chapter, local Cartesian co-ordinate system s-n, which is u-v shown in Figure 5-30, is used, where $\hat{\mathbf{s}}$ and $\hat{\mathbf{n}}$ are the respective unit vectors. The former is normal from the surface of the contact surface and the latter is tangent to the surface. Conventional Penalty method will be explained in this section.

Contact is indicated when the contact node K penetrates the target surface defined by the effective thickness projected from node I, J and K with Equations (5-71). The magnitude of penetration is represented as $p$ and calculated with the following Equation for node-surface contact.

$$p = \begin{cases} t_{\text{eff,i}} - (\mathbf{u}_{\text{K}} - \mathbf{u}_{\text{I}})^{\text{T}} \hat{\mathbf{n}}, & \text{if } (\mathbf{u}_{\text{K}} - \mathbf{u}_{\text{I}})^{\text{T}} \hat{\mathbf{n}} > 0 \\ t_{\text{eff,o}} + (\mathbf{u}_{\text{K}} - \mathbf{u}_{\text{I}})^{\text{T}} \hat{\mathbf{n}}, & \text{if } (\mathbf{u}_{\text{K}} - \mathbf{u}_{\text{I}})^{\text{T}} \hat{\mathbf{n}} \le 0 \end{cases} \tag{5-82}$$

For node-node contact situation, $p$ should be calculated by using Equation (5-83) with different $\hat{\mathbf{s}}$ and $\hat{\mathbf{n}}$.

$$p = t_{\text{eff,i}} + (t_{\text{eff,o}} - t_{\text{eff,i}}) \left( \frac{\tan^{-1}\left( \dfrac{-(\mathbf{u}_{\text{K}} - \mathbf{u}_{\text{I/J}})^{\text{T}} \hat{\mathbf{s}}}{(\mathbf{u}_{\text{K}} - \mathbf{u}_{\text{I/J}})^{\text{T}} \hat{\mathbf{n}}} \right)}{\pi} \right) - (\mathbf{u}_{\text{K}} - \mathbf{u}_{\text{I/J}})^{\text{T}} \hat{\mathbf{n}} \tag{5-83}$$

where $\mathbf{u}_{\text{I}}$ and $\mathbf{u}_{\text{K}}$ are the global displacement of node I, node K respectively. $\mathbf{u}_{\text{I/J}}$ is the global displacement of either node I or node J depending on the node-node contact scenario. The $t_{\text{eff,o}}$ and $t_{\text{eff,i}}$ are the outer effective thickness and the inner effective thickness respectively. The $\hat{\mathbf{s}}$ and $\hat{\mathbf{n}}$ used in Equation (5-83) are calculated as below.

$$\hat{\mathbf{n}} = \frac{\mathbf{u}_{\text{K}} - \mathbf{u}_{\text{I/J}}}{|\mathbf{u}_{\text{K}} - \mathbf{u}_{\text{I/J}}|} \tag{5-84}$$

$$\hat{\mathbf{s}} = \hat{\mathbf{\upsilon}} \times \hat{\mathbf{n}} \tag{5-85}$$

where $\hat{\mathbf{\upsilon}}$ is the unit normal vector to the global x-y plane.

Penalty method consists of placing normal interface springs between all penetrating nodes and the contact surface. The normal contact force is predicted using the following equation by penalty method.

$$F_n = \begin{cases} K_n p & \text{if } p > 0 \\ 0 & \text{if } p \le 0 \end{cases} \tag{5-86}$$

where $K_n$ is the user-defined contact stiffness and $p$ is the penetration of the contact node, which is defined by Equations (5-82) and (5-83). The direction of $F_n$ requires to be manipulated accordingly. Stable computational time step size is unaffected by the existence of the interfaces. If the contact pressure or force is too large, it may cause unacceptable element penetration, where the contact node will penetrate through the thickness. Scaling up the contact stiffness and/or scaling down the time step size can reduce these unrealistic penetrations [102]. Thus, it is presumed that the stable or realistic contact stiffness may have the following relationships:

$$K_{n,cr} \propto F_{n,cr} \tag{5-87}$$

$$K_{n,cr} \propto \frac{1}{dt} \tag{5-88}$$

where $K_{n,cr}$ is the desired contact stiffness, $F_{n,cr}$ is the critical contact force and $dt$ is the time step size. If only through penetration (previously referred as element penetration) is concerned, the relationship can then be amended as below.

$$K_{n,cr} \propto \frac{1}{\min(t_{eff,o}, t_{eff,i})} \tag{5-89}$$

where the $\min(t_{eff,o}, t_{eff,i})$ is the minimum value of the outer effective thickness and the inner effective thickness. Numerical examples in the later section will show the effect of different contact stiffness values on the conventional contact-impact penalty method.

## B. Adaptive Penalty (AP) Method

The first two relations above (Equations (5-87) and (5-88)) should be applied. The contact normal force, resultant from the contact-impact, should be great enough to push back significant normal distance, but not causing unrealistic disturbance to the system (may be caused by very high contact stiffness). A virtual node C, defined as the contact point with contact node K of the contact surface along the link consists of node I and node J. At the contacting time step *t*, the following equations are established referring to central difference temporal integration method.

$$\mathrm{d}v^{t}_{\mathrm{K,n}} = \frac{F^{t}_{\mathrm{K,n}}dt}{m_{\mathrm{K}}} \tag{5-90}$$

$$\mathrm{d}v^{t}_{\mathrm{C,n}} = \frac{F^{t}_{\mathrm{C,n}}dt}{m_{\mathrm{C}}} \tag{5-91}$$

where $\mathrm{d}v^{t}_{\mathrm{K,n}}$ and $\mathrm{d}v^{t}_{\mathrm{C,n}}$ are the change of velocity of node K and node C respectively at contacting time step *t* in the direction of n in the local s-n co-ordinate system. The *dt* is the time step size. The nodal mass of node K and node C is represented as $m_{\mathrm{K}}$ and $m_{\mathrm{C}}$ respectively. The $m_{\mathrm{C}}$ is the mass of the target element. Figure 5-30 shows the arrangement of the respective variables mentioned in a schematic diagram. The terms $\mathrm{d}v^{t}_{\mathrm{K,n}}$ and $\mathrm{d}v^{t}_{\mathrm{C,n}}$ are defined by the following equations.

$$\mathrm{d}v^{t}_{\mathrm{K,n}} = v^{t+0.5}_{\mathrm{K,n}} + v^{t-0.5}_{\mathrm{K,n}} \tag{5-92}$$

$$\mathrm{d}v^{t}_{\mathrm{C,n}} = v^{t+0.5}_{\mathrm{C,n}} + v^{t-0.5}_{\mathrm{C,n}} \tag{5-93}$$

where $v^{t+0.5}_{\mathrm{K,n}}$ and $v^{t+0.5}_{\mathrm{C,n}}$ are the nodal velocities of node K and C respectively at time step *t*+0.5. The change of the velocity should be able to push back the contact node K to the contact boundary. Thus, the following relationship is used.

$$\left(\mathrm{d}v^{t}_{\mathrm{K,n}} - \mathrm{d}v^{t}_{\mathrm{C,n}}\right)dt = p \tag{5-94}$$

and,

$$F_{\mathrm{C,n}} = -F_{\mathrm{K,n}} \tag{5-95}$$

From Equations (5-90) to (5-95), the recommended contact stiffness are given as the following.

$$K_{\mathrm{n,AP}} = \frac{m_{\mathrm{K}} m_{\mathrm{C}}}{\left(m_{\mathrm{K}} + m_{\mathrm{C}}\right)dt^{2}} \tag{5-96}$$

Equation (5-96) shows that the contact stiffness recommended is directly proportional to $F_{n,cr}$ and inversely proportional to the square of *dt*, which correlates with Equations (5-87) and (5-88).

In general, the value of *dt* should not be too large where the contact node penetrates through the contact surface at one time step without violating any of the contact-impact conditions.

## C. Kinematics Contact Algorithm (KCA)

In predicting the normal contact-impact response of the contact element, a Kinematics Contact-Impact Algorithm (KCA) is proposed. The KCA is based on kinematics impact, which involves the conservation of momentum, and coefficient of restitution. Node-node contact is idealised as contact-impact of 2 spheres. Node-surface contact is idealised as contact of a sphere (the contact node) onto a flat surface (the target element surface). Constraint control can be applied to handle impractical penetration when subjected to the KCA.

Conservation of momentum is assumed before and after the process of contact-impact. Equation (5-97) is employed for node-surface contact-impact problem.

$$m_K \mathbf{v}_K{}^T \hat{\mathbf{n}} + m_E \mathbf{v}_c{}^T \hat{\mathbf{n}} = m_K \mathbf{v}'_K{}^T \hat{\mathbf{n}} + m_E \mathbf{v}'_c{}^T \hat{\mathbf{n}} \tag{5-97}$$

where $m_K$ and $m_E$ are the lumped mass of the contact node and the mass of the target element respectively. The velocities of the node before and after contact are denoted as $\mathbf{v}_K$ and $\mathbf{v}'_K$. The parameters $\mathbf{v}_c$ and $\mathbf{v}'_c$ are the velocity of the contacting point of the target surface before and after contact. $\mathbf{v}_c$ is derived from Equation (5-98).

$$\mathbf{v}_c = \mathbf{v}_I + (\mathbf{v}_J - \mathbf{v}_I)\left(\frac{(\mathbf{u}_K - \mathbf{u}_I)^T \hat{\mathbf{s}}}{(\mathbf{u}_J - \mathbf{u}_I)^T \hat{\mathbf{s}}}\right) \tag{5-98}$$

The local contact-impact of the contact element is presumed to have a very short period of restitution compared to the time interval used in the temporal integration. The

Coefficient of restitution is then used to describe the response of the contact-impact. Equations (5-99) and (5-100) are derived to obtain the post-impact normal velocity of node K and post-impact normal velocity of the contact point at the target element.

$$v'_{\mathrm{c,n}} = \mathbf{v}_{\mathrm{c}}{}^{\mathrm{T}}\hat{\mathbf{n}}(m_{\mathrm{c}} - m_{\mathrm{K}}e) + m_{\mathrm{K}}\mathbf{v}_{\mathrm{K}}{}^{\mathrm{T}}\hat{\mathbf{n}}\frac{1+\varepsilon}{m_{\mathrm{c}} + m_{\mathrm{K}}} \tag{5-99}$$

$$v'_{\mathrm{K,n}} = v'_{\mathrm{c,n}} - e(\mathbf{v}_{\mathrm{K}} - \mathbf{v}_{\mathrm{c}})^{\mathrm{T}}\hat{\mathbf{n}} \tag{5-100}$$

where $v'_{\mathrm{K,n}}$ and $v'_{\mathrm{c,n}}$ are the normal components (in n direction of the local co-ordinate system) of the contact node velocity and the contact point velocity of the target element respectively. The coefficient of restitution is denoted as *e* in Equations (5-99) and (5-100). For a frictionless contact problem, the post-impact tangent velocities of nodes (nodes I, J and K) and the contact point at the target element remain the same as the pre-impact value.

For node-node contact-impact problem, the momentum conservation equation is changed to,

$$m_{\mathrm{K}}\mathbf{v}_{\mathrm{K}}{}^{\mathrm{T}}\hat{\mathbf{n}} + m_{\mathrm{I/J}}\mathbf{v}_{\mathrm{I/J}}{}^{\mathrm{T}}\hat{\mathbf{n}} = m_{\mathrm{K}}\mathbf{v}'_{\mathrm{K}}{}^{\mathrm{T}}\hat{\mathbf{n}} + m_{\mathrm{I/J}}\mathbf{v}'_{\mathrm{I/J}}{}^{\mathrm{T}}\hat{\mathbf{n}} \tag{5-101}$$

where the I/J subscripts indicate the corresponding value (mass, pre or post impact velocity) for either node I or node J of the contact element, depending on the contacting node of the target element. The unit vector, $\hat{\mathbf{n}}$, is defined as the following.

$$\hat{\mathbf{n}} = \frac{(\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I/J}})}{|\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I/J}}|} \tag{5-102}$$

while the unit vector $\hat{\mathbf{s}}$ is defined as the following when involving node-node contact.

$$\hat{\mathbf{s}} = \hat{\mathbf{\upsilon}} \times \hat{\mathbf{n}} \tag{5-103}$$

where $\hat{\mathbf{\upsilon}}$ is the unit normal vector to the global x-y plane.

The post-impact velocities of respective nodes are calculated with the following formulas.

$$v'_{\mathrm{I/J,n}} = \mathbf{v}_{\mathrm{I/J}}{}^{\mathrm{T}}\hat{\mathbf{n}}(m_{\mathrm{I/J}} - m_{\mathrm{K}}e) + m_{\mathrm{K}}\mathbf{v}_{\mathrm{K}}{}^{\mathrm{T}}\hat{\mathbf{n}}\frac{1+e}{m_{\mathrm{I/J}} + m_{\mathrm{K}}} \qquad (5\text{-}104)$$

$$v'_{\mathrm{K,n}} = v'_{\mathrm{I/J,n}} - e(\mathbf{v}_{\mathrm{K}} - \mathbf{v}_{\mathrm{I/J}})^{\mathrm{T}}\hat{\mathbf{n}} \qquad (5\text{-}105)$$

The KCA may not be capable of bringing the contact node out of the target surface. Permanent offset or penetration may occur especially when setting the coefficient of restitution, $\varepsilon$, to zero. To handle such situation, an additional algorithm called Constraint Control after the KCA execution was developed. Depth of penetration of the contact node, $p$, on the target surface after applying the KCA is calculated as the following.

$$p = \begin{cases} t_{\mathrm{eff,i}} - \left((\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I}})^{\mathrm{T}}\hat{\mathbf{n}} + v'_{\mathrm{K,n}} - v'_{\mathrm{c,n}}\right) & \text{if } (\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I}})^{\mathrm{T}}\hat{\mathbf{n}} > 0 \\ t_{\mathrm{eff,o}} + \left((\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I}})^{\mathrm{T}}\hat{\mathbf{n}} + v'_{\mathrm{K,n}} - v'_{\mathrm{c,n}}\right) & \text{if } (\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I}})^{\mathrm{T}}\hat{\mathbf{n}} \le 0 \end{cases} \qquad (5\text{-}106)$$

For node-node contact situation, $p$ should be calculated by using Equation (5-107) with $\hat{\mathbf{n}}$ from Equation (5-102).

$$p = t_{\mathrm{eff,i}} + (t_{\mathrm{eff,o}} - t_{\mathrm{eff,i}})\left(\frac{\tan^{-1}\left(\dfrac{-(\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I/J}})^{\mathrm{T}}\hat{\mathbf{s}}}{(\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I/J}})^{\mathrm{T}}\hat{\mathbf{n}}}\right)}{\pi}\right) - \left((\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I/J}})^{\mathrm{T}}\hat{\mathbf{n}} + v'_{\mathrm{K,n}} - v'_{\mathrm{I/J,n}}\right) \qquad (5\text{-}107)$$

The parameter $p$ has a positive value if the KCA does not manage to separate the contact from penetration. The velocities of the corresponding nodes are further updated after the KCA. Equations below are used to obtain the final velocity after applying constraint control for node-surface contact-impact.

$$v''_{\mathrm{c,n}} = v'_{\mathrm{c,n}} - p\frac{m_{\mathrm{K}}}{(m_{\mathrm{E}} + m_{\mathrm{K}})\Delta t}\frac{(\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I}})^{\mathrm{T}}\hat{\mathbf{n}}}{\left|(\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I}})^{\mathrm{T}}\hat{\mathbf{n}}\right|} \qquad (5\text{-}108)$$

$$v''_{\mathrm{K,n}} = v'_{\mathrm{K,n}} + p\frac{m_{\mathrm{E}}}{(m_{\mathrm{E}} + m_{\mathrm{K}})\Delta t}\frac{(\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I}})^{\mathrm{T}}\hat{\mathbf{n}}}{\left|(\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I}})^{\mathrm{T}}\hat{\mathbf{n}}\right|} \qquad (5\text{-}109)$$

For node-node contact-impact scenario, the velocities of the contacting nodes are updated with the following equations. All the $\hat{\mathbf{n}}$ involved in node-node contact is derived from Equation (5-102).

$$v''_{\mathrm{I/J,n}} = v'_{\mathrm{I/J,n}} - p\frac{m_{\mathrm{K}}}{(m_{\mathrm{I/J}} + m_{\mathrm{K}})\Delta t}\frac{(\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I/J}})^{\mathrm{T}}\hat{\mathbf{n}}}{\left|(\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I/J}})^{\mathrm{T}}\hat{\mathbf{n}}\right|} \tag{5-110}$$

$$v''_{\mathrm{K,n}} = v'_{\mathrm{K,n}} + p\frac{m_{\mathrm{I/J}}}{(m_{\mathrm{I/J}} + m_{\mathrm{K}})\Delta t}\frac{(\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I/J}})^{\mathrm{T}}\hat{\mathbf{n}}}{\left|(\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I/J}})^{\mathrm{T}}\hat{\mathbf{n}}\right|} \tag{5-111}$$

where subscripted "I/J" denotes the corresponding values of either node I or node J, and depends on the contacting node.

## D. KCA/AP method

The KCA/AP method is a combination of Kinematics Contact-Impact Algorithm and Adaptive Penalty method. This takes the advantage of both KCA and AP methods, where the pre-impact velocities of the contact nodes and resultant penetration magnitude are considered simultaneously. The KCA alone may face permanent or steady state penetration. This phenomenon may occur with the KCA, especially when the coefficient of restitution is set to 0 and long contact time is involved. The KCA also shows its incapability to handle contacting problems when the displacement and velocities of the contact surface element and/or contact node is/are manipulated externally. In other words, they are forced to change dramatically not because of the internal property of the material. Typical example would be setting a physical constraint such as setting rotational limit of a pin joint. The KCA does not consider the computational magnitude of the penetration. The combination of KCA/AP adds a sense to the magnitude of contact penetration. Comparing to AP, KCA/AP takes care of the pre-impact situation. The KCA/AP is having the following relations.

$$F_{\mathrm{C,n}} = -F_{\mathrm{K,n}} = \frac{v'_{\mathrm{C,n}} - v_{\mathrm{C,n}}}{dt\ m_C} - K_{n,AP}\,p \tag{5-112}$$

Equation (5-112) is for the case where the contact node is above the contact surface in the local n-s co-ordinate system. The sign of "$-K_{n,AP}p$" has to be inverted for the reverse case.

### 5.10.4 RELEASING CONDITIONS

Releasing conditions are conditions to be satisfied when the contact node is considered releasing from the contact-impact. The most important condition is when the contact node is out of the contact region (the effective thickness in this case). The following is the main contact condition where $p$ is the magnitude of penetration of the contact node onto the contact surface.

$$p \leq 0 \tag{5-113}$$

Either of the following additional contact conditions is valid when the contact node is leaving the contact surface.

$$v_{\mathrm{K,n}} - v_{\mathrm{C,n}} > 0 \quad \text{if } (\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I}})^{\mathrm{T}} \hat{\mathbf{n}} > 0 \tag{5-114}$$

$$v_{\mathrm{K,n}} - v_{\mathrm{C,n}} < 0 \quad \text{if } (\mathbf{u}_{\mathrm{K}} - \mathbf{u}_{\mathrm{I}})^{\mathrm{T}} \hat{\mathbf{n}} < 0 \tag{5-115}$$

Two alternative sets of releasing conditions can be used for the Penalty method and the AP method. They are consisting of condition in Equation (5-113) alone and consisting of conditions in Equations (5-113), (5-114), and (5-115). The former will ensure energy conservation for the Penalty method and the AP method. But, it will generate periodical contact-impact response or oscillations towards the system. Both the KCA and KCA/AP can only use the latter set of releasing conditions.

### 5.10.5 TANGENT CONTACT-IMPACT RESPONSE

Classical Coulomb's friction model is used to predict the tangential response when subjecting to contact-impact situation. Micro-slip phenomenon is ignored in rigid Coulomb's friction model. Tangential shear towards the contact elements is not considered in this work for simplicity purpose. The effect can be included by redefining location of the frictional force acting on the surface of the element of the structure.

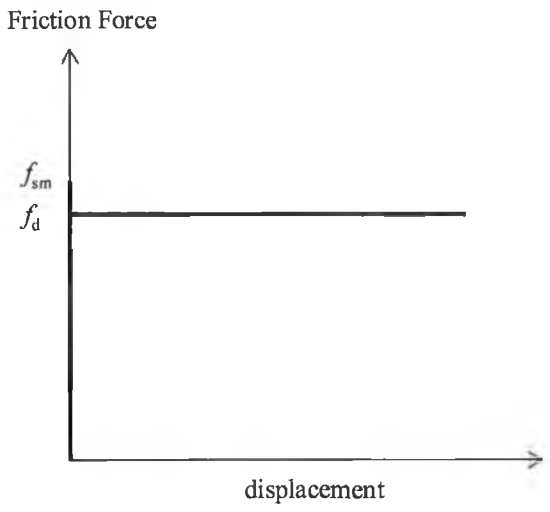


Figure 5-35: Rigid Coulomb's friction model

Figure 5-35 shows the classical frictional response chart. The parameters $f_{\mathrm{sm}}$ and $f_{\mathrm{d}}$ are the static friction limit and dynamic friction force respectively. $f_{\mathrm{sm}}$ is defined by Equation (5-116). The relation between $f_{\mathrm{sm}}$ and $f_{\mathrm{d}}$ is defined by the dynamic/static friction factor, $f_{\mathrm{d/s}}$.

$$f_{\mathrm{sm}} = \mu F_{\mathrm{n}} \tag{5-116}$$

$$f_{\mathrm{d/s}} = \frac{f_{\mathrm{d}}}{f_{\mathrm{sm}}} \tag{5-117}$$

where $\mu$ is the coefficient of static friction limit and $F_{\mathrm{n}}$ is the contact force, which is acting normal to the surface of the target surface. $F_{\mathrm{n}}$ is derived from the change of contact element velocity.

$$F_{\mathrm{n}} = m_K \frac{v''_{\mathrm{K,n}} - \mathbf{v}_{\mathrm{K}}{}^{\mathrm{T}} \hat{\mathbf{n}}}{\Delta t} \tag{5-118}$$

Where $\Delta t$ is the time interval of the temporal integration. The tangential friction force is derived as the following.

$$F_s = \begin{cases} -m_K \dfrac{\mathbf{v}_K{}^T\hat{\mathbf{s}}}{dt}, & \text{if } \left| m_K \dfrac{\mathbf{v}_K{}^T\hat{\mathbf{s}}}{dt} \right| \le f_{sm} \\ -f_{d/s} f_{sm} \dfrac{\mathbf{v}_K{}^T\hat{\mathbf{s}}}{\left|\mathbf{v}_K{}^T\hat{\mathbf{s}}\right|}, & \text{if } \left| m_K \dfrac{\mathbf{v}_K{}^T\hat{\mathbf{s}}}{dt} \right| > f_{sm} \end{cases} \tag{5-119}$$

Sticking occurs where the friction force, $F_s$, is capable of holding the contact node from moving. Sliding occurs when the force is beyond the static friction limit. Sticking and sliding conditions are the former and latter conditions of Equation (5-119).

### 5.10.6 INTEGRATION OF CONTACT ELEMENTS

After the consideration of the normal and tangential contact-impact responses, the nodal forces of the contact element are updated. As mentioned, tangential shear effect due to the contact thickness is not considered in this work for simplicity purpose.

The response throughout of an independent node-surface contact element is derived from the following equations.

$$\mathbf{F}'_I = \mathbf{F}_I + \frac{(\mathbf{u}_K - \mathbf{u}_I)^T\hat{\mathbf{s}}}{(\mathbf{u}_J - \mathbf{u}_I)^T\hat{\mathbf{s}}}(F_n\hat{\mathbf{n}} + F_s\hat{\mathbf{s}}) \tag{5-120}$$

$$\mathbf{F}'_J = \mathbf{F}_J + \left(1 - \frac{(\mathbf{u}_K - \mathbf{u}_I)^T\hat{\mathbf{s}}}{(\mathbf{u}_J - \mathbf{u}_I)^T\hat{\mathbf{s}}}\right)(F_n\hat{\mathbf{n}} + F_s\hat{\mathbf{s}}) \tag{5-121}$$

$$\mathbf{F}'_K = \mathbf{F}_K + F_n\hat{\mathbf{n}} + F_s\hat{\mathbf{s}} \tag{5-122}$$

where $\mathbf{F}'_I$, $\mathbf{F}'_J$, $\mathbf{F}'_K$, $\mathbf{F}_I$, $\mathbf{F}_J$ and $\mathbf{F}_K$ are the post-impact nodal force and the pre-impact nodal force of node I, J and K respectively.

For node-node contact, Equations (5-120) and (5-121) cannot be used. Instead, Equation (5-123) is used to find the target node I or node J.

$$\mathbf{F}'_{I/J} = \mathbf{F}_{I/J} - F_n\hat{\mathbf{n}} - F_s\hat{\mathbf{s}} \tag{5-123}$$

A contact node has the potential to be on contact with several target elements. On the other hand, a target surface may be subjected to several contact nodes. Thus, proper integration among the common contact elements is required. The common here is referring to two elements having either the same contact node or the same contact surface. Direct superimposition was the first idea that came across. The direct superimposition response of a node after impact is calculated from the following equation.

$$\mathbf{F}_i' = \mathbf{F}_i + \sum_{j=\text{contact element}} \mathbf{Fc}_{i,j} \tag{5-124}$$

$$\mathbf{Fc}_{i,j} = \begin{cases} \left.\dfrac{(\mathbf{u}_K - \mathbf{u}_I)^T \hat{\mathbf{s}}}{(\mathbf{u}_J - \mathbf{u}_I)^T \hat{\mathbf{s}}}(F_n \hat{\mathbf{n}} + F_s \hat{\mathbf{s}})\right|_{\text{contact element } j}, & \text{if node } i \text{ is node I} \\ \left.\left(1 - \dfrac{(\mathbf{u}_K - \mathbf{u}_I)^T \hat{\mathbf{s}}}{(\mathbf{u}_J - \mathbf{u}_I)^T \hat{\mathbf{s}}}\right)(F_n \hat{\mathbf{n}} + F_s \hat{\mathbf{s}})\right|_{\text{contact element } j}, & \text{if node } i \text{ is node J} \\ \left. F_n \hat{\mathbf{n}} + F_s \hat{\mathbf{s}}\right|_{\text{contact element j}}, & \text{if node } i \text{ is node K} \\ 0 \end{cases} \tag{5-125}$$

where $\mathbf{F}_i'$ and $\mathbf{F}_i$ are the post-impact and the pre-impact global force vector of node $i$, and will be used in calculating the acceleration when considering dynamic equilibrium. Generation of extra energy is observed by using this method. Energy conservation superimposition is then suggested. Equation (5-124) is amended as the following.

$$\mathbf{F}_i' = \mathbf{F}_i + \sqrt{\left|\sum \frac{\mathbf{Fc}_{i,j}^{\;3}}{\left|\mathbf{Fc}_{i,j}\right|}\right|} \frac{\sum \dfrac{\mathbf{Fc}_{i,j}^{\;3}}{\left|\mathbf{Fc}_{i,j}\right|}}{\left|\sum \dfrac{\mathbf{Fc}_{i,j}^{\;3}}{\left|\mathbf{Fc}_{i,j}\right|}\right|} \tag{5-126}$$

## 5.10.7 NUMERICAL COMPARISONS OF THE METHODS

Contact-impact responses at element level for methods with different contact parameters are studied. The element modelling is shown in Figure 5-36. The contact node is a mass node with 0.1kg in mass and 0.5m in diameter. The contact surface element is a link with the mass of 3.5kg and dimension of 1x0.5x0.3m. The dimensions and mass of the model were selected arbitrarily. Thus, the contact effective thickness, $t_{eff}$, is equal to 0.75m. A constant force, F, of 20N is acting from the top vertically at the contact node. Tangential response or friction is not involved. The models are subjected to different contact-impact configurations. Table 5-4 shows the configuration of contact models used in this section.

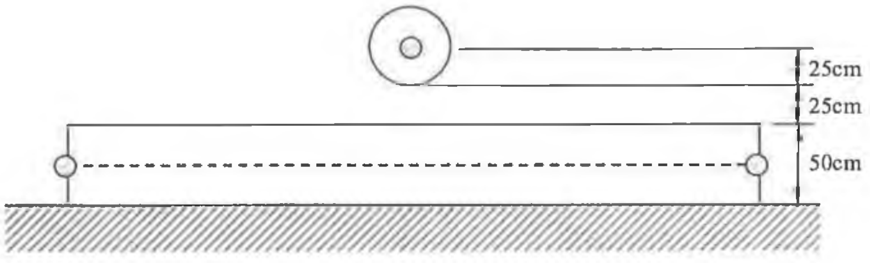


Figure 5-36: Contact-impact of a mass node onto a surface

Table 5-4: Models' configuration

| Model Code | Contact-Impact Model | Time Step Size (μs) | Contact Parameters | Releasing Conditions (equation no.) |
|---|---|---|---|---|
| P1B | Penalty | 50 | $K_n$=500 | (5-113),(5-114), (5-115) |
| P2B | Penalty | 50 | $K_n$=50000 | (5-113),(5-114), (5-115) |
| P3B | Penalty | 50 | $K_n$=50x10$^6$ | (5-113),(5-114), (5-115) |
| P4B | Penalty | 50 | $K_n$=50x10$^9$ | (5-113),(5-114), (5-115) |
| AP1B | Adaptive Penalty | 50 | | (5-113),(5-114), (5-115) |
| P2A | Penalty | 50 | $K_n$=50000 | (5-113) |
| AP1A | Adaptive Penalty | 50 | | (5-113) |
| K1B | KCA | 50 | $\varepsilon$=0.0 | (5-113),(5-114), (5-115) |
| K2B | KCA | 50 | $\varepsilon$=0.5 | (5-113),(5-114), (5-115) |
| K3B | KCA | 50 | $\varepsilon$=1.0 | (5-113),(5-114), (5-115) |
| KP1B | KCA/AP | 50 | $\varepsilon$=0.0 | (5-113),(5-114), (5-115) |
| KP2B | KCA/AP | 50 | $\varepsilon$=0.5 | (5-113),(5-114), (5-115) |
| KP3B | KCA/AP | 50 | $\varepsilon$=1.0 | (5-113),(5-114), (5-115) |
| AP2B | Adaptive Penalty | 5 | | (5-113),(5-114), (5-115) |
| AP3B | Adaptive Penalty | 0.5 | | (5-113),(5-114), (5-115) |
| AP4B | Adaptive Penalty | 0.05 | | (5-113),(5-114), (5-115) |

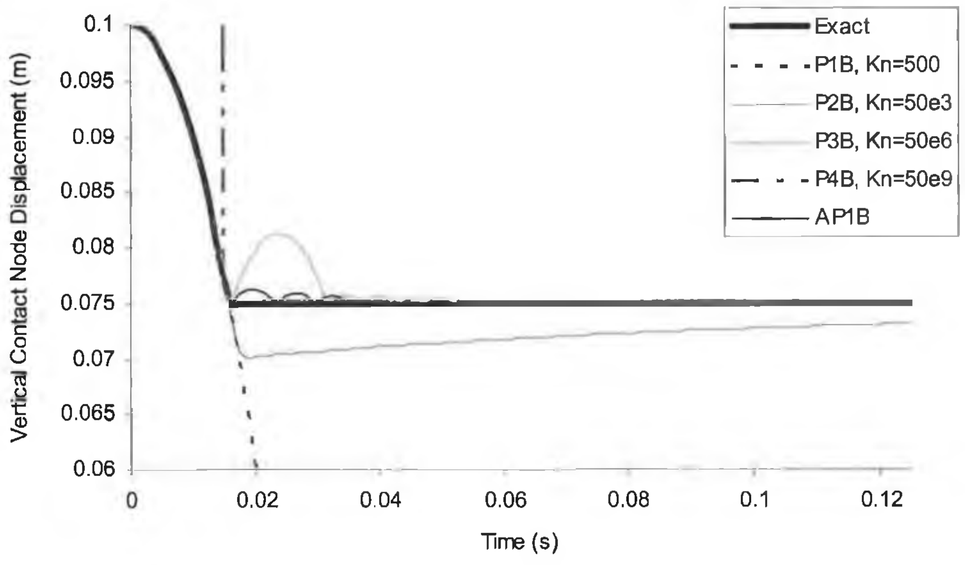


Figure 5-37: Penalty method and AP method contact-impact responses with different contact stiffness

Figure 5-37 shows the comparison of vertical response between the Penalty Method and the Adaptive Penalty method with different contact stiffness values. Vertical displacement histories of the contact nodal centre for different contact models are shown. Contact stiffness is selected arbitrarily. Releasing conditions consist of Equations (5-113), (5-114), and (5-115) are used. High contact stiffness will cause over response towards the system, which is showed in model P4B of Figure 5-37. The contact node rebounds more than 100m before reaching its maximum height. On the other hand, small contact stiffness will lead to complete penetration of the contact node through the contact surface. The user has to decide in selecting the proper contact stiffness value, and most of the time the selection is made arbitrarily at the initial stage and fine tuned through trial and error by adjusting the contact stiffness and the time step size. Most of the time, penetration may occur according to model P2B and rebounding may occur according to model P3B are of common selection range. Both models predict unrealistic responses. Model P2B shows initial penetration with a magnitude of more than 0.005m out of 0.075m and gradually recovering to its steady or permanent penetration. The recovery takes more than 0.1 second in this particular case, which is too long in explicit coding for computational methods. With the P3B model, the contact node rebounds and revisits the contact surface several times before resting on the contact surface. The initial rebounding magnitude is slightly more than 0.005m. The rebounding/revisiting phenomenon lasts for 0.045 second. The Adaptive Penalty method gives a satisfactory result with minor rebounding magnitude of 0.0013m and rebounding time of 0.02 second. Significant revisiting cycle number is only 3 compared to 6 with model P3B. The AP method is capable of resting the contact node on the contact surface while providing required resistive normal force to the contact node after 3 times revisiting. Thus, the AP method gives satisfactory results and no user-defined parameter is required. This also shows that the use of Penalty method needs a realistic contact stiffness, which is normally defined by trial-and-error.

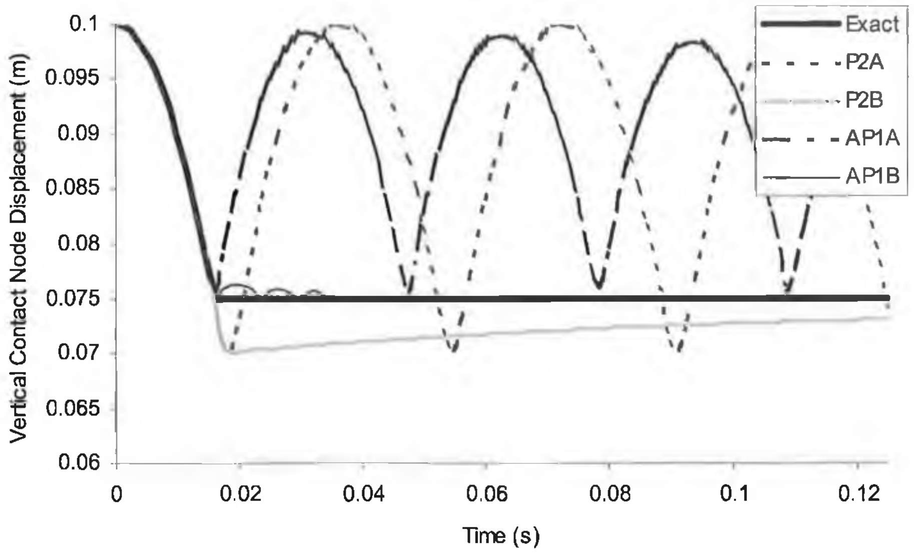


Figure 5-38: Contact-impact for different releasing conditions

Figure 5-38 compares the two different sets of releasing conditions described in section 5.10.4. The simple releasing condition (Equation (5-113)) caused semi-sinusoidal displacement cycles whereas the latter dies off the displacement response and reaches the saturation level. When contacting time is long, the latter set of releasing conditions (Equations (5-113), (5-114), and (5-115)) would be more suitable. While in some other cases where perfect elastic impacts are desired at element level (which is not the case normally), the former releasing condition (Equation (5-113)) should be used. For revisiting situation, lesser revisiting cycle is desired as resting indicates that contact element in equilibrium. Besides that, the contact-damping effect can be incorporated with the former releasing condition (Equation (5-113)) to damp out revisiting cycles. Viscous contact damping has been added in many commercial finite element solvers in handling contact and contact-impact problem, especially involving long contact time such as metal forming, etc [102].

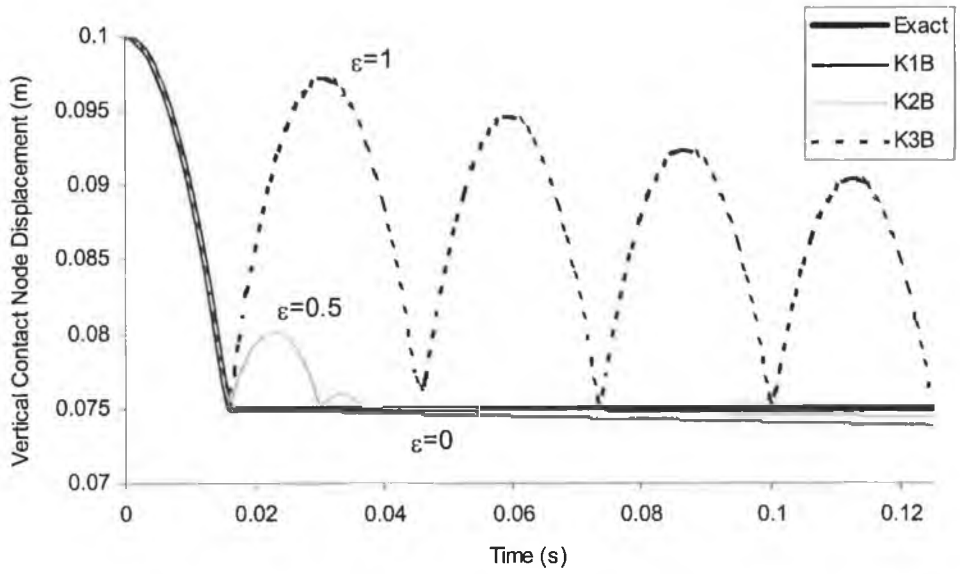


Figure 5-39: KCA contact-impact responses with different coefficient of restitution

Responses with different coefficients of restitution using the KCA are shown in Figure 5-39. The K3B model is a perfect elastic impact, where the coefficient of restitution is equal to 1. The decrease of the vertical displacement peak after each cycle of contact is because of the nodal compensation of the contact surface. The nodes are restricted in vertical movement. The K2B model shows 2 significant rebounds after the initial impact then dies off. The K1B model is a perfect plastic impact, where the coefficient of restitution equals to 0. The response does not show any rebound. Both models suffer gradual permanent penetration after resting on the surface. This is again due to the nodal compensation. The KCA naturally does not take account of the depth of penetration and leads to the unexpected permanent penetration. The AP may be combined with the KCA. Figure 5-40 shows the combined KCA/AP responses with different coefficients of restitution. The KCA/AP with perfect plastic impact shows better response compare to the AP method. The KP1B model shows a rebound, which is near to the magnitude of the AP1B's first rebound, and dies off resting on the contact surface. On the other hand, AP1B needs 3 significant rebounds. The KCA/AP with the coefficient of restitution equals to 1 and 0.5 show unrest rebounds within the time history. The KP3B rebounding magnitude is too large and additional kinetic energy is obviously produced. Thus, it is not recommended. The KP2B's bouncing backs do not show any constant pattern over the studied time history. In short the KCA/AP with local perfect plastic impact is recommended, but not others.

The time step size sensitivity of the AP is studied briefly. Figure 5-41 shows vertical responses with respective size of the time step. The critical time step size for the system is more than 80 microseconds. The chart shows the reduction of significant revisiting cycle from 3 (AP1B) to 1 (AP2B) when reducing the time step size from 50 microseconds to 5 microseconds. The magnitude of the AP2B peak is reduced comparing to AP1B's. When further reducing the time step size to 0.5 microseconds, which is less than 1% of the allowable time step size, the magnitude of the peak increases to the magnitude of 0.087m. Only 2 significant revisiting cycles occur. With 50 nanoseconds as the time step size, the peak only increases to 0.0874m with 2 significant revisiting cycles. This shows, further reducing time step size will have only small increment in the magnitude of the peak and probably having 2 revisiting cycles before resting on the contact surface. All the responses are within acceptable range.

As the whole, the AP method is capable of replacing the existence penalty method. It is able to predict a good contact stiffness. The KCA/AP method shows a better result with perfect plastic local impact, compared to AP method. With different releasing conditions, both the AP and KCA/AP methods are capable of damping out the revisiting cycles efficiently.

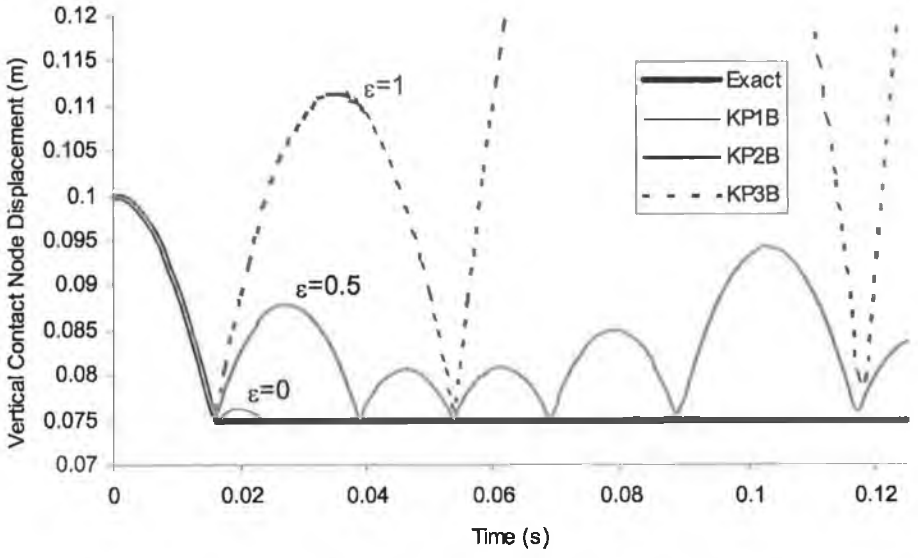


Figure 5-40: KCA/AP contact-impact responses with different coefficient of restitution

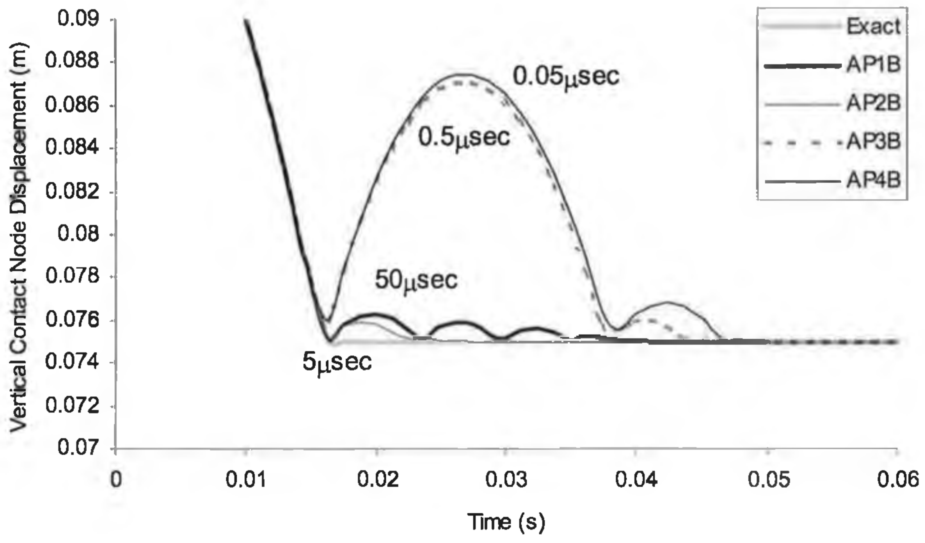


Figure 5-41: AP contact-impact responses with different time step sizes

### 5.10.8 NUMERICAL EXAMPLE 1: CONTACT-IMPACT OF TWO ELASTIC RODS

The case of 2 rods moving with equal initial velocity, $v_o$ in opposition directions is simulated. Figure 5-42 shows the configuration and parameters of the set-up.

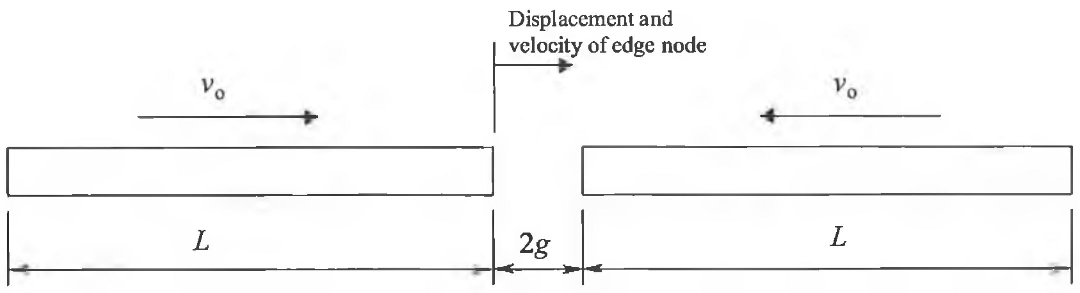


Figure 5-42: Impact of two elastic rods

Table 5-5: Parameters used in example one

| Parameter | Value |
|---|---|
| Young's modulus, $E$ | 30 GN/m$^2$ |
| Density, $\rho$ | 1875 kg/m$^3$ |
| Rod length, $L$ | 1 m |
| Semi-gap, $g$ | 0.05 m |
| Initial velocity, $v_o$ | 100 m/s |
| Rod thickness | 2cm |

The rods are initially undeformed and the problem is symmetric about the point at the impact point of the rods. The rods have pure linear elastic property. The exact solution for the displacement and velocity at the contact surface of the left rod is given as the following.

$$\left.\begin{aligned} u_t &= v_o t \\ v_t &= v_o \end{aligned}\right\} \; t \le t_{\text{impact}} \tag{5-127}$$

$$\left.\begin{aligned} u_t &= g \\ v_t &= 0 \end{aligned}\right\} \; t_{\text{impact}} < t \le t_{\text{release}} \tag{5-128}$$

$$\left.\begin{aligned} u_t &= 2g + 2v_o L\sqrt{\rho/E} - v_o t \\ v_t &= -v_o \end{aligned}\right\} \quad t > t_{\text{release}} \tag{5-129}$$

$$\begin{aligned} t_{\text{impact}} &= g/v_o \\ t_{\text{release}} &= g/v_o + 2L\sqrt{\rho/E} \end{aligned} \tag{5-130}$$

For computations, each rod is discretised by 10 equal length elements and the mass is lumped. From the theory, the cross-section of the rod will not affect the impact response. The values of the parameter are as shown in Table 5-5.

Table 5-6: Simulation models' configuration for example one

| Model Code | Time Interval, $\Delta t$ (μsec) | Coefficient of Restitution, $\varepsilon$ | Constraint Control |
|---|---|---|---|
| K1N | 0.5 | 1.0 | No |
| K2N | 0.5 | 0.5 | No |
| K3N | 0.5 | 0.0 | No |
| K3Y | 0.5 | 0.0 | Yes |
| K4Y | 1.0 | 0.0 | Yes |
| K5Y | 3.0 | 0.0 | Yes |
| K6Y | 6.0 | 0.0 | Yes |

Different KCA parameters are used for separate simulation to study the characteristics of the algorithm. Table 5-6 shows the configuration of KCA for different models. Figure 5-43 shows a displacement history chart of the contact surface of the rod for $\varepsilon =$ 1.0, 0.5 and 0.0. The model K1N shows the best correlation to the exact solution compared to the others. It is particularly true as the exact solution assumes a perfect elastic collision. In the other hand, the model K3N shows maximum difference. Figure 5-44 shows an enlarged portion of Figure 5-43. The figure shows that the KCA with $\varepsilon =$ 1.0 does not show a stable displacement during the period of restitution. At the other extreme with $\varepsilon = 0$, the contact node penetrates the target node (thickness) at a constant rate. This can be predicted from the theory. As with the example, the velocities of the contact node and the target node after the first impact will be zeros. The contact node will penetrate further when the second wave of impact arrived. The normal impact response (displacement) will still be zero. For $\varepsilon = 0.5$ the result shows very good correlation compared to the exact solution for this particular problem during the period of restitution. Figure 5-45 shows the end node (contact surface of the rod) velocity history of the respective models. As predicted, the local elastic contact ($\varepsilon$ = 1.0) condition will lead to an unstable region during the period of restitution.

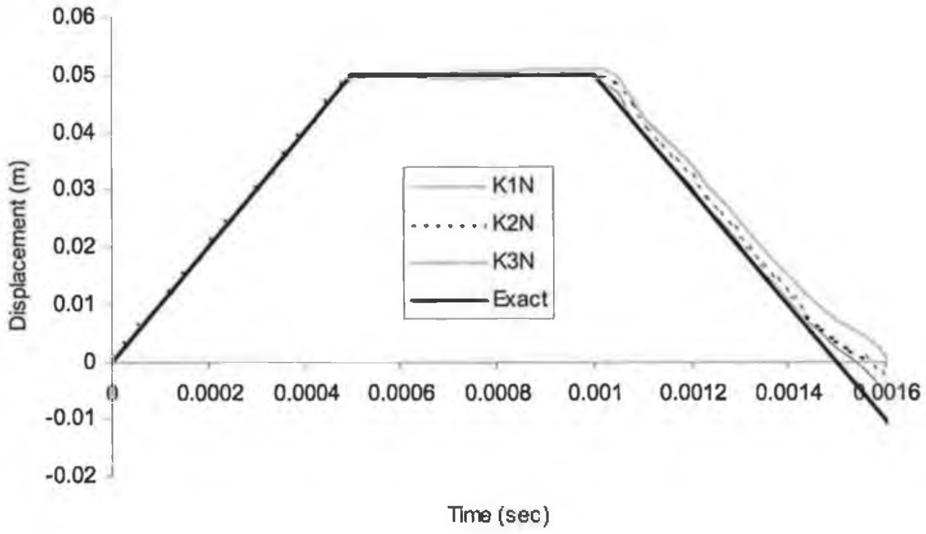


Figure 5-43: Displacement history of the rod contact surface with KCA

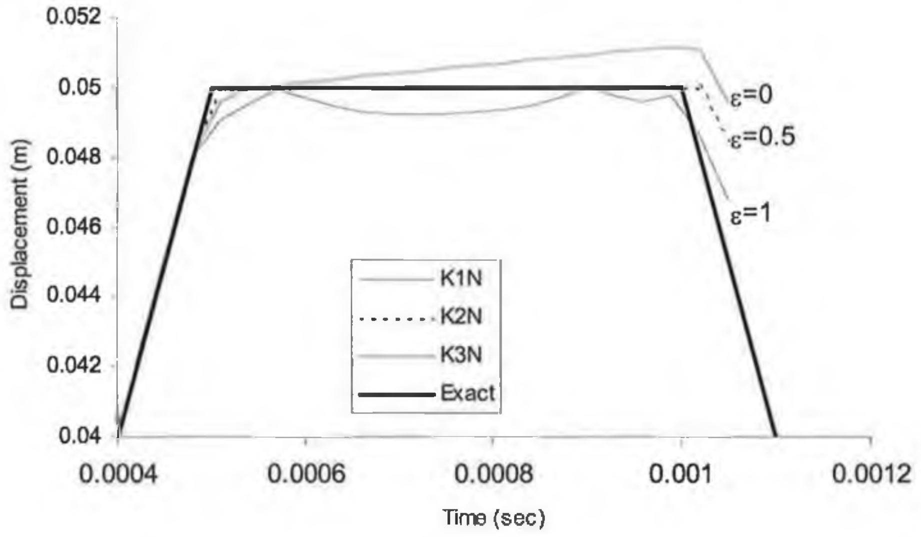


Figure 5-44: Enlarged displacement history of the rod contact surface with KCA

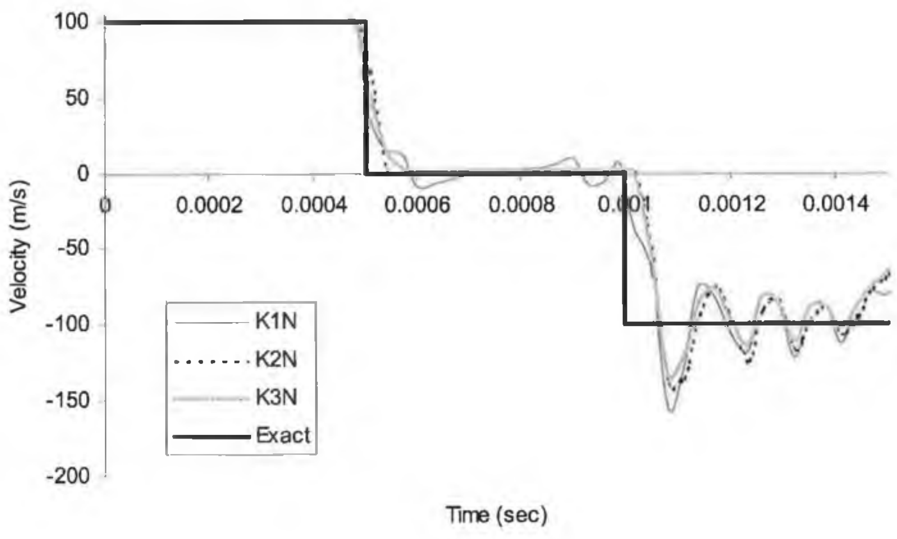


Figure 5-45: Velocity history of the rod contact surface with KCA

Constraint control has been applied to avoid the local inelastic contact ($\varepsilon = 0$) which causes permanent penetration locally. Figure 5-46 shows the end rod displacement with and without constraint control. The Constraint control algorithm is capable of offsetting the unwanted penetration onto the contact surface. In addition, the predicted result with constraint control algorithm shows better agreement when compared to the exact solution. The constraint control is particular useful when contact node is subjected to a geometry constrained target surface. Figure 5-47 shows the velocity history of both with and without constraint control algorithm.

Different time step sizes ranging from 0.5μsec to 6μsec are chosen to perform the simulation. Figure 5-48 shows the displacement history with different sizes of time step. Step size variance does not affect significantly the performance of calculation unless the time step size is too large for which numerical instability will occur. Numerical instability is shown in Figure 5-48 with time step equal to 6 μsec. The configurations of the example was set in such a way that the critical KCA time step is smaller than that of the structure calculated with Courant-Friedrichs-Levy criterion. The critical time step of the rod in this example is $2.25\text{x}10^{-5}$ second. The detail sensitivity analysis of the time interval is beyond scope of this study.

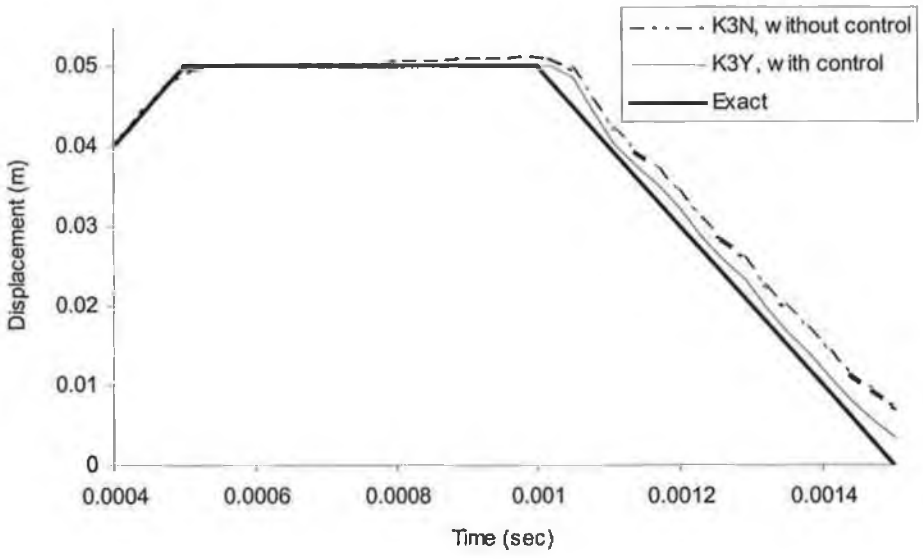


Figure 5-46: Displacement history with KCA constraint control

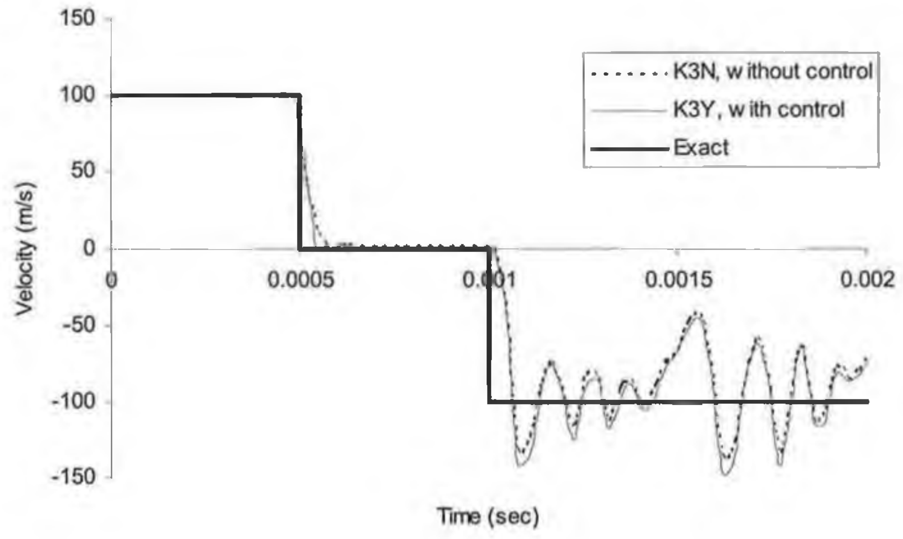


Figure 5-47: Velocity history with KCA constraint control

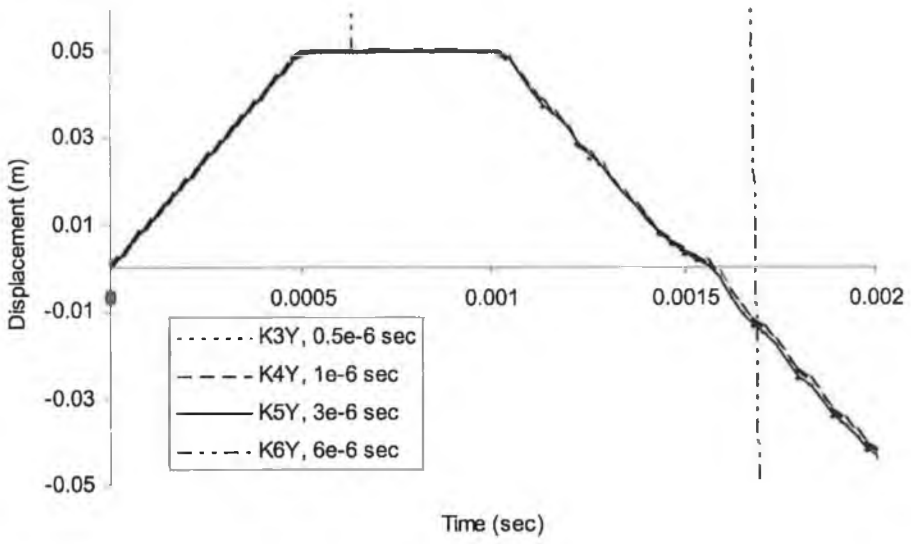


Figure 5-48: KCA Displacement history with different time step sizes

The AP method and the perfect plastic KCA/AP method are involved respectively and the horizontal displacement histories of the edge node are shown in Figure 5-49. Both use releasing conditions of Equations (5-113), (5-114), and (5-115). In comparison with the exact solution, both the AP and the KCA/AP methods show good correlation. The AP method shows an initial rebound, which required 2 revisiting cycles to settle down on the contact surface. No significant revisiting cycle occurs with the KCA/AP. Post impact displacement profile with both the AP and the KCA/AP methods are roughly similar.

Figure 5-50 shows instantaneous velocity history profiles. Instantaneous velocity is referred to the velocity of the node at a particular time instance *t* irrespective of neither the overall displacement nor the overall velocity. Thus, the instantaneous value is fluctuating, not stable in general and not suitable to represent neither the real velocity nor the real acceleration. Fluctuating velocity profiles of both the AP and the KCA/AP responses are as expected and demonstrated in Figure 5-50. A big jump occurs with the AP method at the initial stage of the impact but not the KCA/AP method.

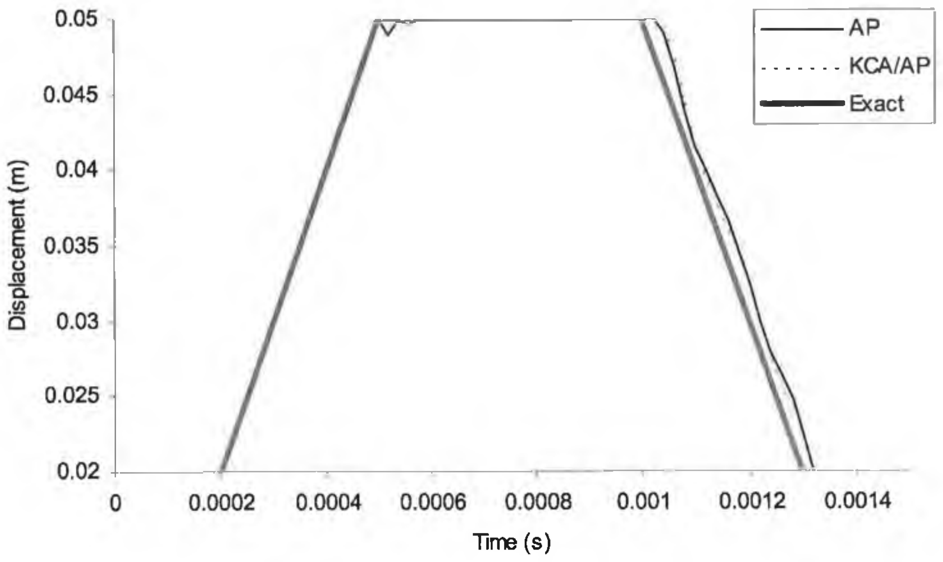


Figure 5-49: Displacement history of the rod contact surface with AP and KCA/AP methods

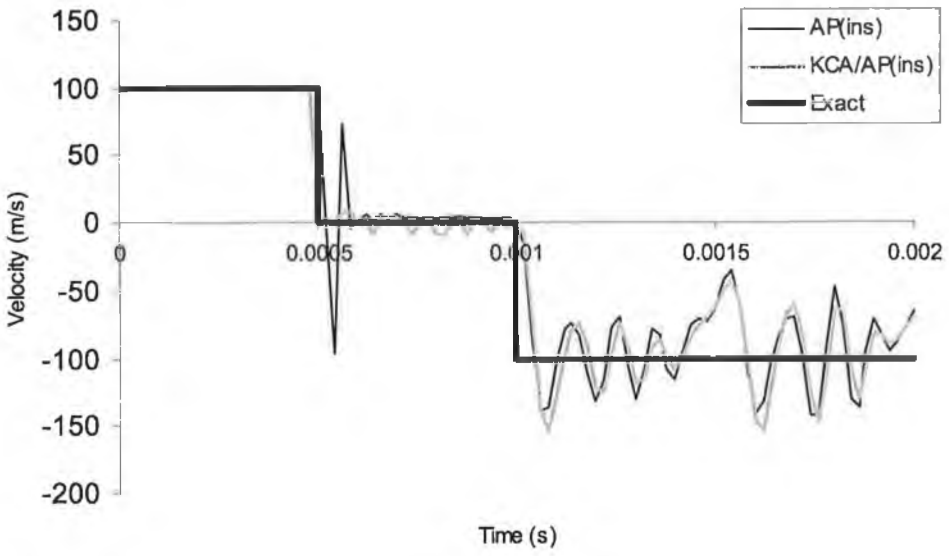


Figure 5-50: Instantaneous velocity history of the rod contact surface with AP and KCA/AP methods

Least square method (LSM) has been incorporated in calculating the effective velocities. Figure 5-51 shows LSM velocity profiles for the AP and the KCA/AP with *n* = 7. AP method demonstrates two revisiting cycles in the velocity profile, which has the same number of cycles in displacement profile (refer to Figure 5-50). The KCA/AP method does not show any significant revisiting cycle. Both the AP and the KCA/AP show good correlation with the exact solution.

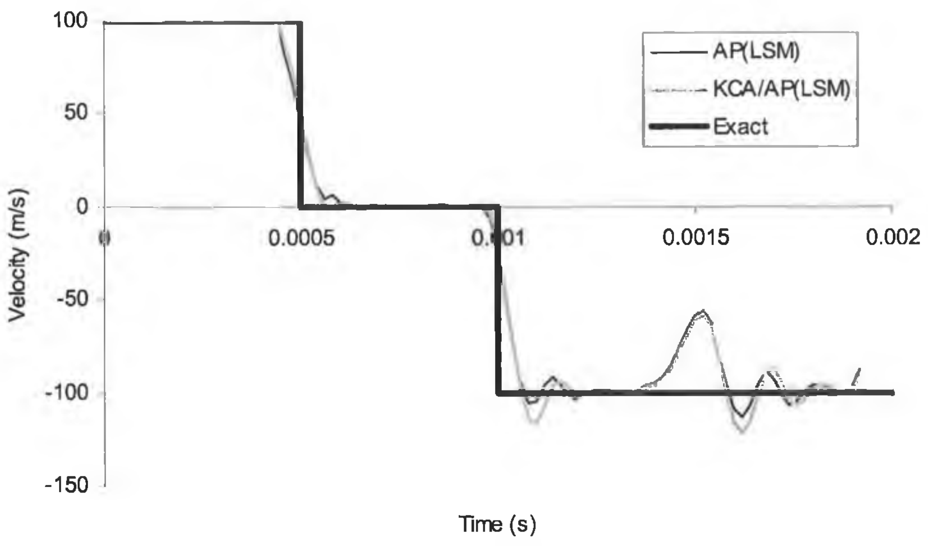


Figure 5-51: LSM velocity history of the rod contact surface with AP and KCA/AP methods

## 5.10.9 NUMERICAL EXAMPLE 2: BAR MOVEMENT WITH FRICTION

A bar with 1m x 0.1m x 0.1m in dimension is placed on a solid ground. The coefficient of friction between the bar and the ground is 0.5 and the dynamic/static factor is 0.9. The bar is subjected to an initial velocity of 0.0220725m/s horizontally. The bar is under frictional resistance due to the weight of the bar and the gravitational acceleration is 9.81m/s$^2$. Figure 5-52 shows the initial setting of the example and the discretised 1-D elements. The exact solution of the rod velocity along the ground is given by the following equations.

$$v_t = v_0 - \mu f_{d/s} g t \qquad (5\text{-}131)$$

where $\mu$ is the coefficient of friction and $f_{d/s}$ is the dynamic/static factor.

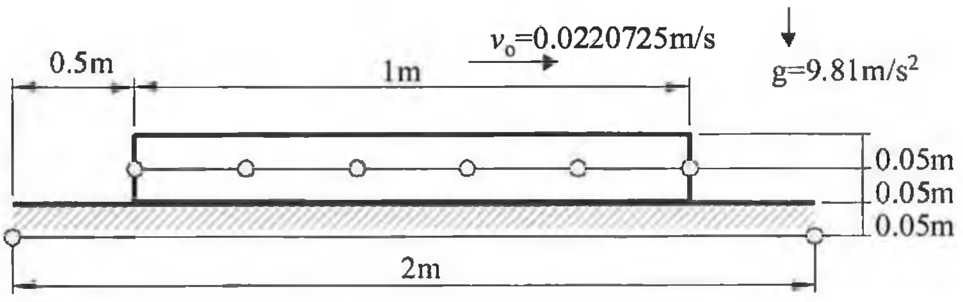


Figure 5-52: Rod movement with friction problem

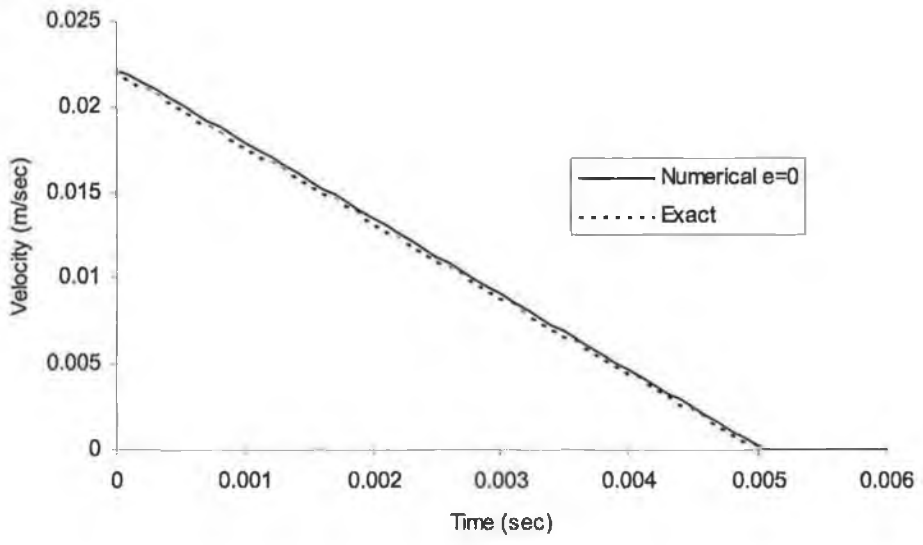


Figure 5-53: Velocity history of rod friction problem with KCA

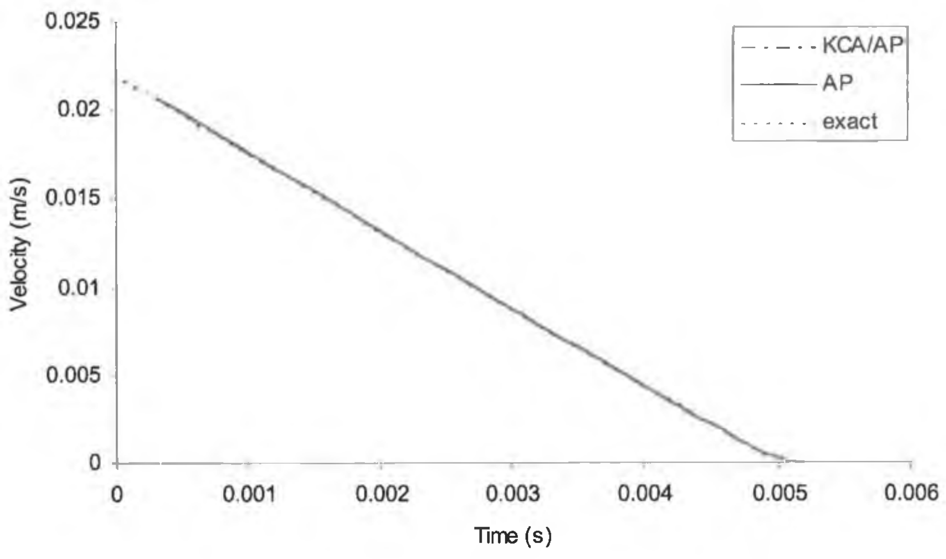


Figure 5-54: Velocity history of rod friction problem with AP and KCA/AP

The bar is predicted to stop moving at time t=0.005sec. Figure 5-53 shows the velocity history according to both the numerical and the exact solutions. The predicted velocity history is in excellent agreement with the exact solution. The contact algorithm used is perfect plastic KCA with constraint control.

Figure 5-54 shows the velocity history according to both the numerical and the exact solutions. The predicted average LSM velocities are in excellent agreement with the exact solution. The perfect plastic KCA/AP method shows similar correlation when compared to the AP method.

### 5.10.10 NUMERICAL EXAMPLE 3: CONTACT-IMPACT OF OPPOSITE-SIDE DOUBLE CANTILEVERS

A relatively complex problem is illustrated in this section. The problem involves non-linear large deformation. Figure 5-55 shows the arrangement of two cantilevers with end fixed at opposite sides. The cross-section of each cantilever is 0.05m x 0.01m and the density is 7860kg/m$^3$. The upper and the lower cantilevers are discretised with 10 and 15 nodes (9 and 14 elements) respectively. The numerical parameters are shown in Table 5-7. The nearest 3 nodes of the lower rod outer edge are subjected to an initial velocity of 20m/s each. The material model is shown in Figure 5-56.

Figure 5-57 shows the deformed profiles with the time interval between each profile equal to 2msec. The figure is a cropped diagram which shows only the middle contact region from the whole length of both the cantilevers. The problem has been simulated until t=30msec. Figure 5-58 shows the second half of the cantilevers' response where secondary impact starts. Figure 5-59 shows the vertical displacement history of the cantilevers' free edge. This example illustrates the capability of the MSS contact-impact algorithm.

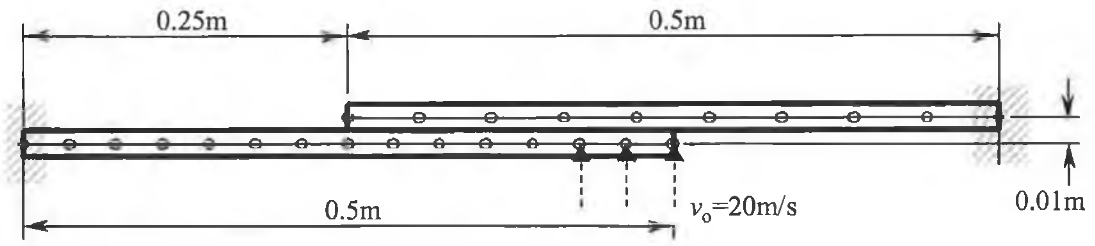


Figure 5-55: Opposite-side double-cantilever setting

Table 5-7: Numerical parameters for example three

| Parameter | Value |
|---|---|
| Contact type | KCA with Constraint Control |
| Friction type | Rigid Coulomb's Friction Model |
| Effective thickness, $t_{eff}$ | 0.01m |
| Coefficient of friction, $\mu$ | 0.2 |
| Dynamic/static factor | 0.9 |
| Coefficient of restitution, $\varepsilon$ | 0 |
| Time step, $\Delta t$ | 1μsec |

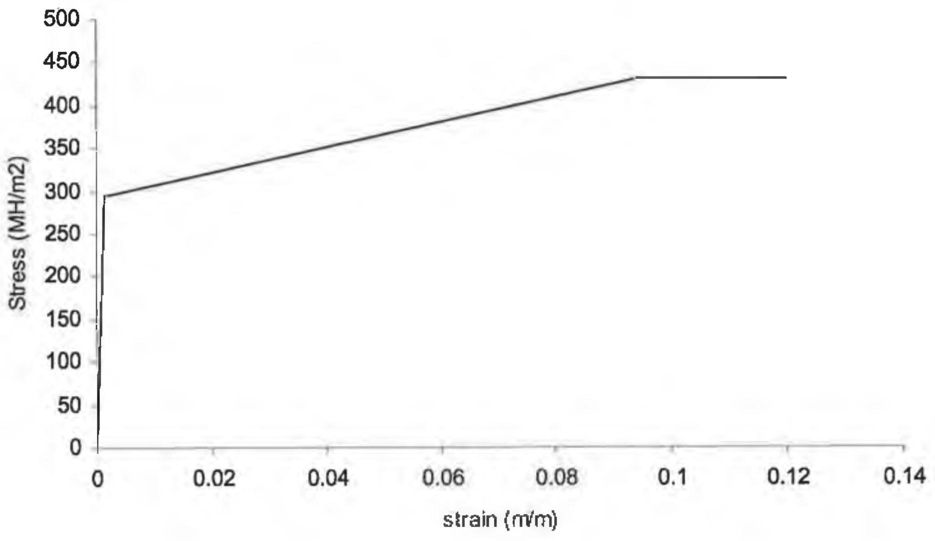


Figure 5-56: Idealised material stress-strain relation

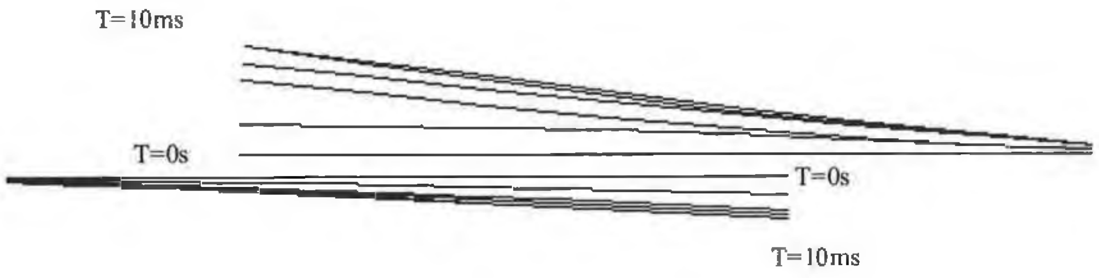


Figure 5-57: Cropped cantilever profiles

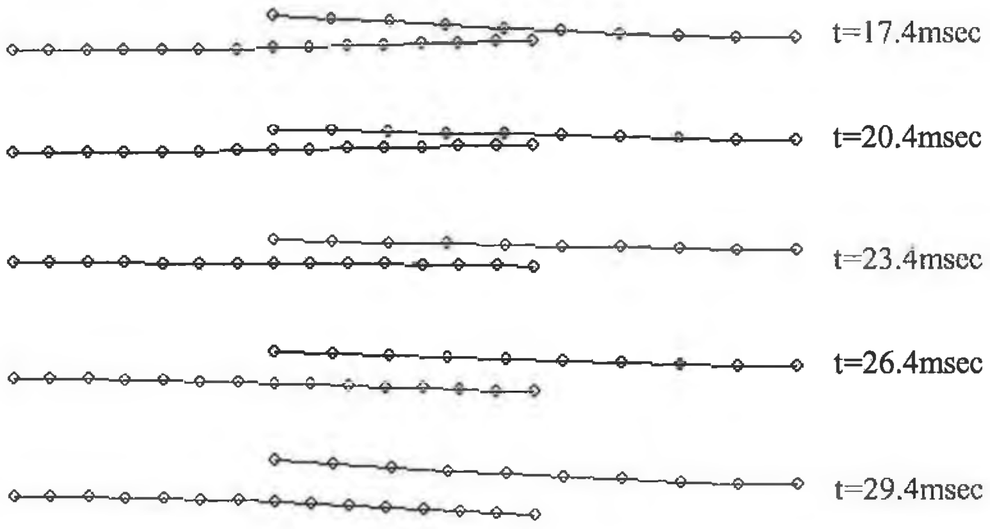


Figure 5-58: Cantilever transient responses

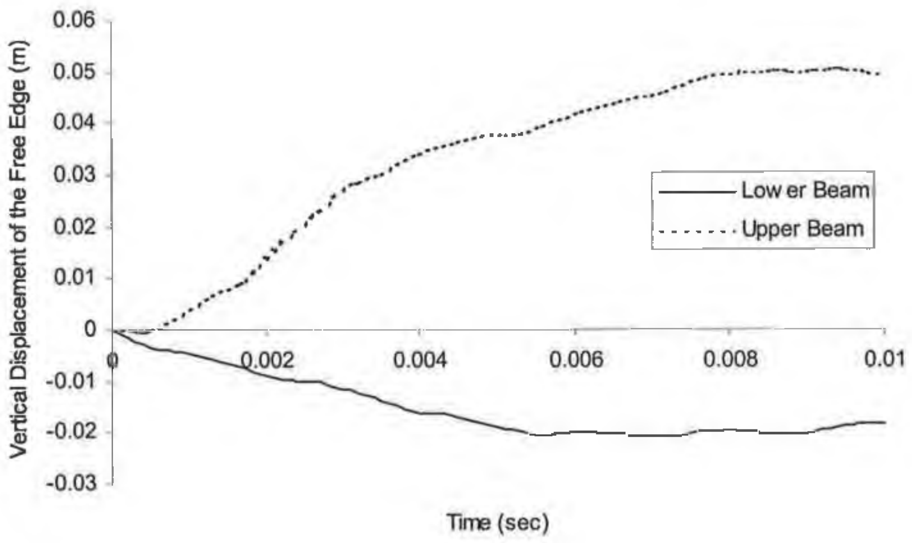


Figure 5-59: Vertical displacement history of the free edges

### 5.10.11 NUMERICAL EXAMPLE 4: CONTACT-IMPACT OF SAME-SIDE DOUBLE CANTILEVERS

To further illustrate the capability of the MSS contact-impact algorithms, an example of double cantilevers subjected to contact-impact are considered. Figure 5-60 shows the schematic diagram of the system. Both beams are discretised by 20 nodes (19 elements). The nodal numbering system starts from 0 instead of 1. Three nodes at the edge of the upper beam are subjected to an initial velocity of $v_o$. The configuration of the system is shown in Table 5-8. The system is subjected to Coulomb's friction model and strain rate sensitivity model of Cowper and Symbonds.

The stress-strain relationship is shown in Figure 5-61. Figure 5-62 shows the transient response of both the beams by using the AP method. Figure 5-63 shows the displacement history of node 19 (edge node of the lower beam) and node 39 (edge node of the upper beam). They are labelled as AP-19, AP-39, KCA/AP-19 and KCA/AP-39 for node 19 with AP method, node 39 with AP method, node 19 with KCA/AP method and node 39 with KCA/AP method respectively. The AP shows slower response than the KCA/AP, which is expected (refer section 5.10.7). In comparing the computational efforts, the KCA/AP requires 79 seconds to complete 7500 time steps with a Pentium II 300MHz machine, while the AP requires 76 seconds.

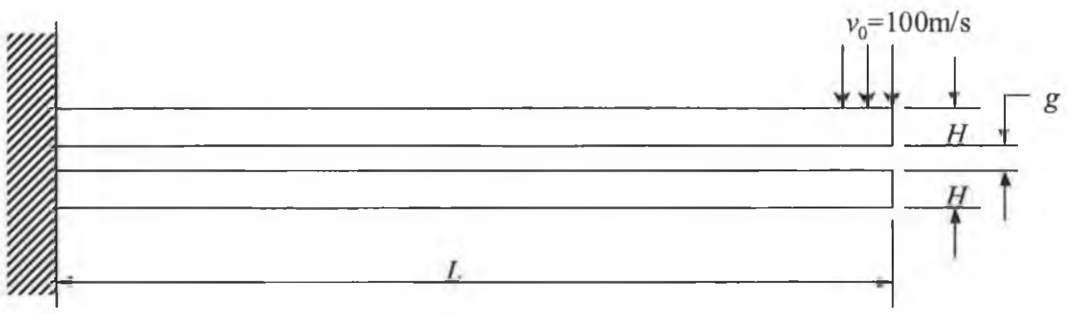


Figure 5-60: Same-side double-cantilever setting

Table 5-8: Parameters used in same-side double cantilevers problem

| Parameter | Value |
|---|---|
| Density, $\rho$ | 1875 kg/m$^3$ |
| Rod length, $L$ | 1 m |
| Gap, $g$ | 0.01 m |
| Initial velocity, $v_0$ | 100 m/s |
| Rod thickness | 0.1m |
| Rod Width | 0.1m |
| $C$ | 1000s$^{-1}$ |
| $P$ | 0.65 |
| $\mu$ | 0.15 |
| $f_{d/s}$ | 0.9 |

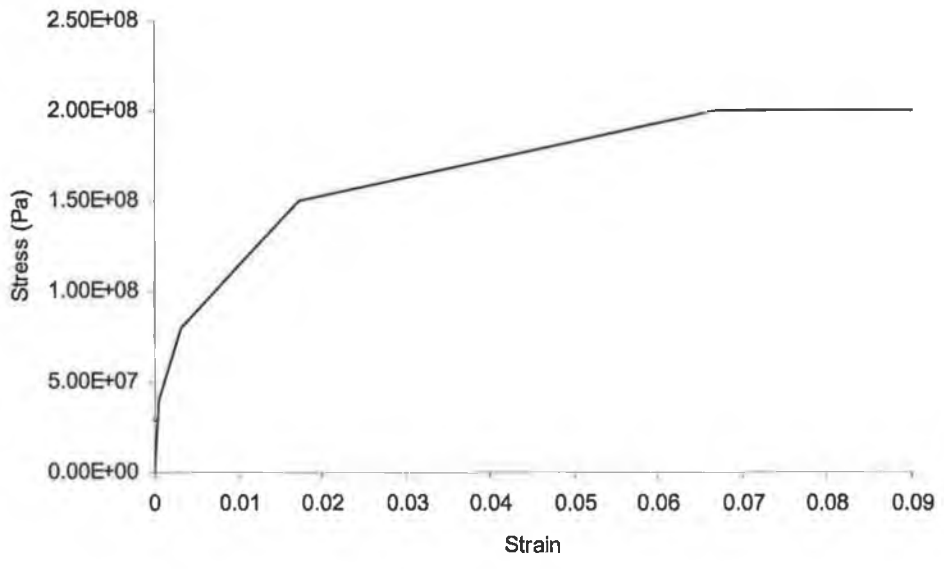


Figure 5-61: Idealised material stress-strain relation

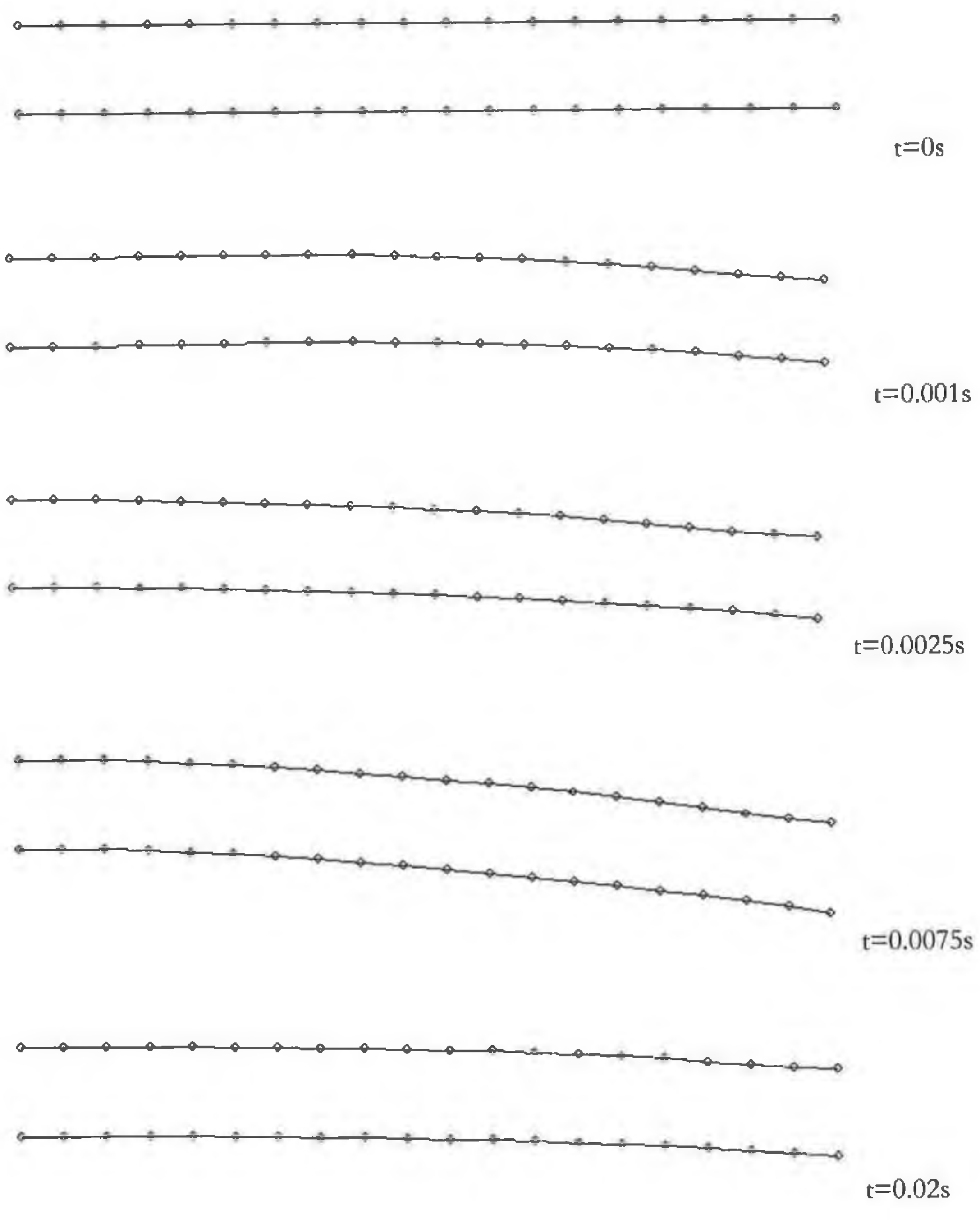


Figure 5-62: Cantilever transient responses

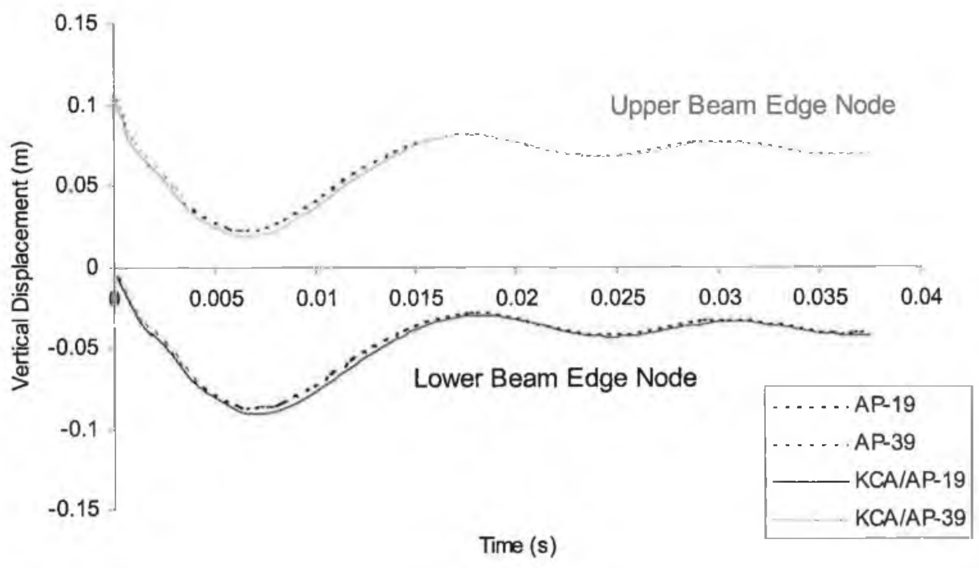


Figure 5-63: Edge nodes displacement profile

## 5.10.12 DISCUSSIONS AND SUMMARIES

This section described a complete contact-impact strategy from contact-impact model to the novel kinematics contact-impact algorithm. Constraint control, contact searching and incorporation of friction model are presented. Four numerical examples are shown. Comparisons among different contact-impact handling methods with the exact solution have been done.

The proposed contact-impact algorithms are adequate for most general situations. To cope with more critical contact scenarios, a 2-pass procedure may be implemented easily. The method is capable of handling 1-D model with thickness consideration and beam element as well.

The new contact searching strategy has been presented in the present section. The contact searching scheme has more realistic representation compared to the present general formulations, especially when dealing with the potential voids and overlaps at the intersection of contact elements. The searching strategy is capable of differentiating node-node contact to node-surface contact.

The Adaptive Penalty (AP) method has been illustrated in this section. Besides that, incorporation of the AP with Kinematics Contact Algorithm (KCA) has been described. KCA and conventional Penalty method have been presented. Numerical examples have shown the capability and feasibility of the AP and the KCA/AP in solving numerical contact-impact problems.

The normal contact-impact response can be predicted using the newly developed methods. Different from the existing general contact algorithm such as the penalty method, a user needs to select the contact properties in term of the nature of contact, from pure elastic contact-impact to pure plastic contact-impact for the KCA and nothing for the AP. For KCA with $\varepsilon = 0$ is normally used. For penalty method, penalty spring force law is needed, where the user has to know the proper contact stiffness and etc. The question is often asked about how are the parameters that define the contact force law to be determined. Use of Lagrange multipliers inevitably introduces implicitness [114].

Plastic (partial or pure) contact-impact with the KCA method will develop unwanted local penetration at the contact element. To avoid such local phenomena, constraint control is applied. The numerical examples show the capability of constraint control in bringing the contact node back to the surface of the contact. The situation will be different if element penetration is required.

Contact-impact responses at element level of Penalty method, AP method, KCA and combined KCA/AP methods are studied. Different parametric values have been used. The AP method shows its capability of providing a considerably good contact stiffness especially compared to wild selection. The KCA/AP with perfect plastic impact method shows better contact response compared to the AP method. Both the AP and KCA/AP methods are capable of damping out the revisiting cycles or impact oscillation, which normally achieved with adding viscous damping effects at the contact-impact element.

The KCA/AP is more efficient in damping oscillations compared to the AP. But, the KCA/AP requires additional computational efforts compared to the AP method.

The AP method and the KCA/AP method are capable of replacing the existing Penalty method, which requires user-defined parameter. The contact stiffness of the Penalty method is usually defined by trial and error. The KCA/AP with perfect plastic impact is a better alternative, but the KCA/AP with perfect elastic impact will cause instability especially involving long contact time.

Only minor adjustment is required to work with implicit temporal integration scheme for both the AP and the KCA/AP method. In general, the KCA/AP method with perfect plastic impact and the AP method should be considered as better alternatives for the Penalty method and others in handling numerical contact-impact problems with FE or FD techniques.

Rigid Coulomb's friction model has been incorporated successfully with the methods developed in this work. Unconventional friction model can be incorporated easily. Integration method used in this work does not create additional kinetic energy when multiple contact nodes contacting a target surface or a contact node is in contact with several target surfaces. Numerical examples have been illustrated in this section, which shows the capabilities of the KCA, AP, and KCA/AP methods, as well as the whole contact strategy.

## 5.11 REVOLUTE JOINT ELEMENT

A novel revolute model proposed and developed in this study is capable in handling rotation limits, locking, resistive torque and viscous damping effect. The new model can handle additional situation such as constant resistive torque at a particular range of angle and having contact stiffness at the other. Rotation limits and locking effect are handled with an adaptive rotation spring stiffness developed in this study, which the user is not required to prescribe the spring stiffness. Several other possible models are also proposed. They are forced-displacement method, momentum-balanced method, and energy-conservation method. The Forced-displacement method is a common method for handling physical constraints. The Momentum-balanced method is a common method for predicting the overall post-contact response. A new way of implementing the Momentum-balanced method at element level is described. A new alternative method called Energy Conservation method is shown in this section. Comparisons among these methods are made and for simplicity purpose, torsional effect is not included in this section and 2-D space is used. Node-Layer structural model is used while solving the numerical examples. Central difference temporal integration scheme is applied.

This section explains the new revolute element's component as the first step. Then, formulation of the adaptive stiffness model is explicated. Other possible methods are presented as well. The formulations with locking mechanism, resistive torque and damping are explained then. Contact, locking and releasing conditions are illustrated. Furthermore, computational stability of the revolute joint element is discussed. The author shows the comparison among the Penalty-like method, Adaptive Stiffness method, Momentum-balanced method, Forced-displacement method, and Energy Conservation method. Respective pros, cons and limitation are discussed. A simple numerical example is solved for comparison purposes. Last but not least, a numerical example is simulated and comparisons with both experimental and mathematical result from literature [147] are made.

### 5.11.1 ELEMENT STRUCTURE

The new revolute element has similar basic structure to the ANSYS COMBIN7 revolute joint element (Figure 2-4 of Chapter 2). Table 5-9 shows the primary variables and their brief description used in the new revolute element. $\theta$ is the rotary location of body 2 (represented by Node J and Node L) relative to body 1 (represented by Node I and Node K) about the joint. The valid range of $\theta$ is from 0 to 360 degree (excluding 360 degree) and being calculated by the system with the location of nodes I, J, K and L. Incremental rotary displacement (Eulerian characteristics) can be used, which only for nodes I and J are applicable, and the valid range of $\theta$ is from -180 to 180 degree.

The Node I and Node J are at coincident position for ideal situation where no linear displacement is allowed in between the two nodes. Selection of nodes K and L as reference is assumed to form a rotary rigid reference towards the nodes I and J, means the line formed by nodes I-K and nodes J-L should not undergo any rotary deformation (bending in most cases) about the other axes, besides the revolute axis. For 3D-space element, an additional reference node is required. $I_1$ and $I_2$ are the local rotary or moment inertia of the respective connecting elements. Figure 5-64 shows a typical situation of the two connected bodies.

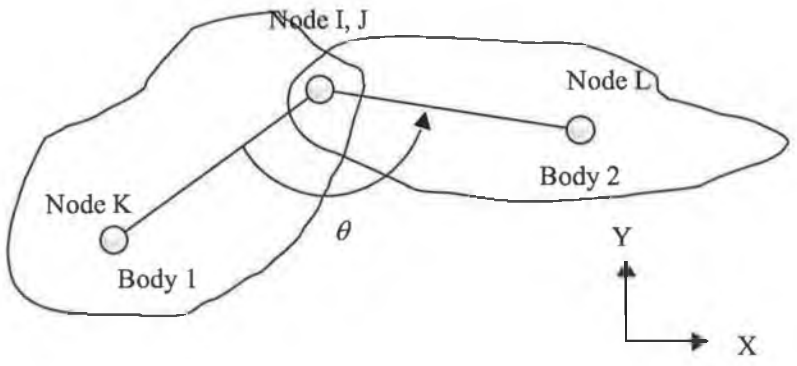


Figure 5-64: General configuration of revolute joint element

Table 5-9: Revolute element structure

| Variable | Description |
|---|---|
| $\theta$ | Rotational position (0-360 degree) |
| $I_1$, $I_2$ | Moment (rotary) inertia of body 1 and body 2 |
| Node I, J | Connecting nodes |
| Node K,L | Reference nodes of node I and J respectively |
| $\theta_i$, $\Gamma_i$ | Array of rotation position and respective resistive torque |
| $\theta_f$ | Rotational forward limit |
| $\theta_r$ | Rotational reverse limit |
| *Block* | Locking flag |
| *C* | Coefficient of viscous damping |

Compared to the ANSYS COMBIN 7 revolute joint element, the *Tf* in Figure 2-4 of Chapter 2 is replaced by variable resistive torque with respect to the rotary location. The rotary location-torque array stores the information about the required resistive torque at a particular rotary location.

Two rotary locations are used to define the forward and the reverse rotary limits. They are $\theta_f$ and $\theta_r$ respectively. A flag, *bLock*, needs to be set in order to activate the locking mechanism after the rotary limit. The Locking/rotary limits spring stiffness, *K* in Figure 2-4, is predicted by the Adaptive Stiffness or other method. The conventional Penalty-like method (used by ANSYS COMBIN 7) can still be applied with the present element by defining the above mentioned torque-location array. The locking/rotary limits stiffness may vary accordingly with the new arrangement.

Viscous damping is included as a function of rotary velocity which involving coefficient of damping, *C*. COMBIN7 element only includes viscous friction instead of viscous damping.

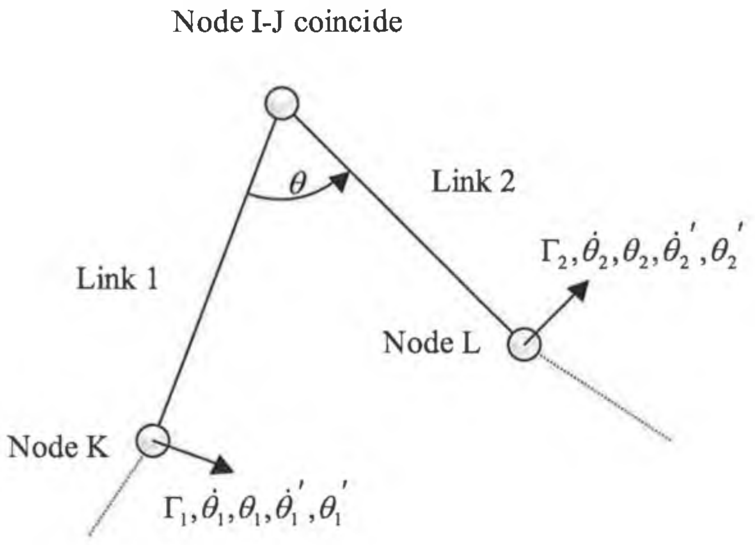

Node I-J coincide
$\theta$
Link 2
$\Gamma_2, \dot{\theta}_2, \theta_2, \dot{\theta}_2', \theta_2'$
Link 1
Node L
Node K
$\Gamma_1, \dot{\theta}_1, \theta_1, \dot{\theta}_1', \theta_1'$

Figure 5-65: Schematic diagram of the revolute element formulations

## 5.11.2 ELEMENT FORMULATIONS

The formulations of the proposed revolute element are explained in this section. The variables shown in Table 5-9 are input variables except the rotary location, $\theta$. The rotary inertia, $I_1$ and $I_2$, can be prescribed or calculated for simple beam element and any 1-D element. For simplification purpose, the formulations in this section are based on 2D space with the revolute joint rotating around the global z-axis. The 3-D space can be achieved with further interpretation of the rotating axis, where a reference node is required. Figure 5-65 is considered for the formulation of the following. The revolute join is placed between 2 link elements, link1 and link 2. Node I and node J of the revolute joint element are at coincident location. The other node of the link elements is chosen as the reference nodes (K and L) of the revolute element. The terms $\Gamma_1$ and $\Gamma_2$ are the resultant torques of the links after reaching rotary limits or locking (used in predicting rotary limit response). $\dot{\theta}_i$ and $\dot{\theta}_i'$ are the rotary velocity immediately before and after reaching rotary the rotary limits of the body $i$ (used in rotary limits validation, locking validation and predict rotary limit response).

### A. Rotary Displacement

The rotary position of body 2 relative to body 1, $\theta$, is calculated by the following equation.

$$\theta = \tan 2^{-1}\left(\frac{\mathbf{u}_r{}^T\hat{\mathbf{Y}}}{\mathbf{u}_r{}^T\hat{\mathbf{X}}}\right) \tag{5-132}$$

where $\mathbf{u}_r$ is the local relative displacement of node L with respective to node K with rotary orientation of node I-J horizon. The $\mathbf{u}_r$ is calculated with the following equation.

$$\mathbf{u}_r = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} (\mathbf{U}_L - \mathbf{U}_J)^T\hat{\mathbf{X}} \\ (\mathbf{U}_L - \mathbf{U}_J)^T\hat{\mathbf{Y}} \\ (\mathbf{U}_L - \mathbf{U}_J)^T\hat{\mathbf{Z}} \end{bmatrix} \tag{5-133}$$

$$\phi = \tan^{-1}\left(\frac{(\mathbf{U}_k - \mathbf{U}_1)^T \hat{\mathbf{Y}}}{(\mathbf{U}_k - \mathbf{U}_1)^T \hat{\mathbf{X}}}\right) \tag{5-134}$$

The $\hat{\mathbf{Y}}$, $\hat{\mathbf{X}}$ and $\hat{\mathbf{Z}}$ are the normal vector of the global Y-axis, X-axis and Z-axis respectively. $\mathbf{U}_i$ is the global displacement vector of node *i*.

## B. Variable Resistive Torque(s)

To cope with variable resistive torque, the value of *Tf* in Figure 2-4 of Chapter 2 varies with respect to the rotary location. The resistive torque-rotary location relation is discretised into finite sets of data and stored in two separate array, $\mathbf{\Gamma}$ and $\boldsymbol{\theta}$. Linear interpolation is applied to find the corresponding resistive torque at any rotary location in between the rotary locations of the array. The corresponding torque, $\Gamma$, at rotary location, $\theta$, is given by the following equation.

$$\Gamma = \sum_{i=0}^{n-1} \begin{cases} \Gamma_i + \dfrac{\theta - \theta_i}{\theta_{i+1} - \theta_i}(\Gamma_{i+1} - \Gamma_i), & \text{if } \theta_i \le \theta < \theta_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{5-135}$$

where *n* is the size of the array starting from zero. Duplicating range of $\theta$ in the array is allowed. Thus, summation function is used in Equation (5-135). For simple case, which is a normal scenario, the array is arranged in such a way that $\theta_i < \theta_{i+1}$, $\theta_0 = 0$ degree and $\theta_n = 360$ degree. The stiffness of the rotary spring at a specific range of rotary displacement can be defined with the following simple relation.

$$K_i = \frac{\Gamma_{i+1} - \Gamma_i}{\theta_{i+1} - \theta_i} \tag{5-136}$$

The formulation can handle the constant resistive torque and/or rotary spring stiffness at the same time. Conventional Penalty-like method handling rotary limits can be achieved with the new formulation of variable *Tf*. A high stiffness can be applied (defined in the formulation of Equation (5-135)) to resist the rotary movement after reaching the rotary limits. Overlapping definition of the resistive torque with rotary displacement is allowed.

## C. Rotary limits and Locking Conditions

Rotary limits consist of forward and reversed limits. The forward and reverse refer to rotary displacement's direction ($\theta$). Forward is anti-clockwise of $\theta$ with positive value in the following formulations and vice versa. Figure 5-66 shows a typical configuration of a revolute element with rotational limits. Limiting conditions are where the rotary displacement of both connected bodies reaches the limits (forward and reverse). Figure 5-67 shows another possible configuration of the rotary limits. $\theta_f$ is larger than $\theta_r$ with the former configuration and vice versa.

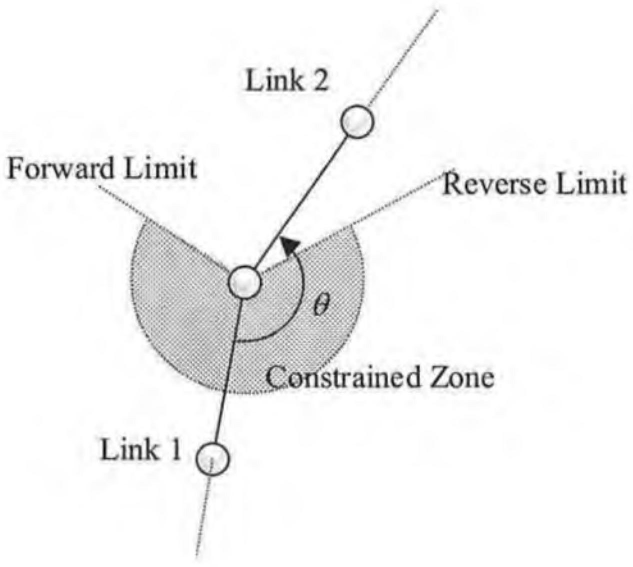


Figure 5-66: Backward constrained revolute joint

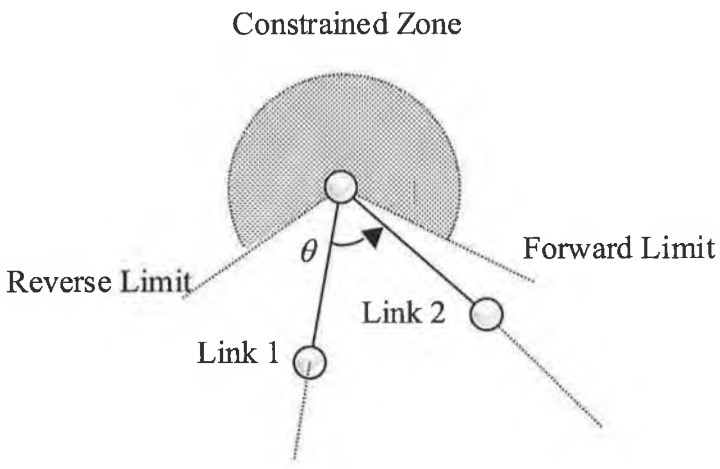


Figure 5-67: Forward constrained revolute joint

For limit is reached, the following conditions have to be met.

$$bLimit = \begin{cases} \text{true}, & \text{if } (\theta_f > \theta_r) \otimes (\theta > \theta_f \oplus \theta < \theta_r) \\ & \text{or if } (\theta_f < \theta_r) \otimes (\theta_f < \theta < \theta_r) \\ \text{false} \end{cases} \tag{5-137}$$

The symbols ⊗ and ⊕ stand for AND and OR Boolean operators respectively. A status flag is used to represent different situation of the revolute element. The status flag, $L_{\text{flag}}$, is defined by the following equation.

$$L_{\text{flag}} = \begin{cases} 0 = \text{free to move} \\ 1 = \text{forward limit reached}, & \text{if } bLimit = \text{true} \otimes \dot{\theta} > 0 \\ 2 = \text{reverse limit reached}, & \text{if } bLimit = \text{true} \otimes \dot{\theta} < 0 \\ 3 = \text{locked}, & \text{if } bLimit = \text{true} \otimes bLock = \text{true} \end{cases} \tag{5-138}$$

where $\dot{\theta}$ is the relative rotary velocity of link 2 to link 1. The *bLock* is a Boolean flag to indicate activation of locking mechanism. The $L_{\text{flag}}$ will be revised with the below releasing conditions provided $L_{\text{flag}}$ is not equal to 3.

$$\text{if } L_{\text{flag}} = 1 \otimes \theta < \theta_f \tag{5-139}$$

$$\text{if } L_{\text{flag}} = 2 \otimes \theta > \theta_r \tag{5-140}$$

Another alternative set of releasing conditions can be used and stated as the following.

$$\text{if } L_{\text{flag}} = 1 \otimes \dot{\theta} < 0 \tag{5-141}$$

$$\text{if } L_{\text{flag}} = 2 \otimes \dot{\theta} > 0 \tag{5-142}$$

The effects of different releasing conditions will be discussed later. The combined validations of rotary limits, locking with the former set of releasing conditions can be stated as the following:

$$L_{\text{flag}} = \begin{cases} 1, & \text{if } bLimit = \text{true} \otimes (L^{\text{p}}_{\text{flag}} = 1 \oplus L^{\text{p}}_{\text{flag}} = 0 \otimes \dot{\theta} > 0) \\ 2, & \text{if } bLimit = \text{true} \otimes (L^{\text{p}}_{\text{flag}} = 2 \oplus L^{\text{p}}_{\text{flag}} = 0 \otimes \dot{\theta} < 0) \\ 3, & \text{if } bLimit = \text{true} \otimes bLock = \text{true} \oplus L^{\text{p}}_{\text{flag}} = 3 \\ 0 \end{cases} \tag{5-143}$$

where $L_{\text{flag}}$ and $L^{\text{p}}_{\text{flag}}$ are the present and the previous status flags. For the later set of releasing conditions, the following is considered.

$$L_{\text{flag}} = \begin{cases} 1, & \text{if } bLimit = \text{true} \otimes (L^{\text{p}}_{\text{flag}} = 1 \oplus L^{\text{p}}_{\text{flag}} = 0) \otimes \dot{\theta} > 0 \\ 2, & \text{if } bLimit = \text{true} \otimes (L^{\text{p}}_{\text{flag}} = 2 \oplus L^{\text{p}}_{\text{flag}} = 0) \otimes \dot{\theta} < 0 \\ 3, & \text{if } bLimit = \text{true} \otimes bLock = \text{true} \oplus L^{\text{p}}_{\text{flag}} = 3 \\ 0 \end{cases} \tag{5-144}$$

## D. Rotary Limit Response Methods

Several methods can be used to predict the post limit response of the connected elements. They are COMBIN7 Penalty-like method, Adaptive stiffness method, Momentum-balanced method, Forced-displacement method and Energy-conservation method.

### D.I Adaptive Stiffness Method

An adaptive method is suggested to predict the rotary limit responses. No user-defined parameter is required. Rotary penetration of connected bodies should be avoided after the rotary limits. Thus the rotary stiffness should be able to push back the connected elements effectively. At the contact time step *t*, the following equations are established referring to central difference temporal integration scheme at the time when the two bodies in are "contact" at the limit boundary.

$$\mathrm{d}\dot{\theta}_1^t = \frac{\Gamma_1^t dt}{I_1} \tag{5-145}$$

$$\mathrm{d}\dot{\theta}_2^t = \frac{\Gamma_2^t dt}{I_2} \tag{5-146}$$

where $\mathrm{d}\dot{\theta}_1^t$ and $\mathrm{d}\dot{\theta}_2^t$ are the change of rotary velocity of bodies (links in this case) 1 and 2 respectively at contacting time step *t*. The *dt* is the time step size. The $\mathrm{d}\dot{\theta}_1^t$ and $\mathrm{d}\dot{\theta}_2^t$ are defined by following equations with central difference temporal integration scheme.

$$\mathrm{d}\dot{\theta}_1^t = \dot{\theta}_1^{t+0.5} - \dot{\theta}_1^{t-0.5} \tag{5-147}$$

$$\mathrm{d}\dot{\theta}_2^t = \dot{\theta}_2^{t+0.5} - \dot{\theta}_2^{t-0.5} \tag{5-148}$$

where $\dot{\theta}_1^{t+0.5}$ and $\dot{\theta}_2^{t+0.5}$ are the rotary velocities of links 1 and 2 respectively at time step *t*+0.5. The change of the velocity should push back the rotating links to the limit contact boundary. Thus, the following relationship is used.

$$\left(\mathrm{d}\dot{\theta}_2^t - \mathrm{d}\dot{\theta}_1^t\right)dt = p \tag{5-149}$$

where *p* is the rotary penetration and calculated with the equation below.

$$p = \begin{cases} \theta_\mathrm{f} - \theta, & \text{if } L_\mathrm{flag} = 1 \\ \theta_\mathrm{r} - \theta, & \text{if } L_\mathrm{flag} = 2 \end{cases} \tag{5-150}$$

Also knowing,

$$\Gamma_1 = -\Gamma_2 \tag{5-151}$$

With the above Equations (5-145) to (5-151), the recommended contact stiffness is given as the following.

$$K_\mathrm{AP} = \frac{I_1 I_2}{\left(I_1 + I_2\right)dt^2} \tag{5-152}$$

Similar to the COMBIN7 Penalty-like method, the magnitude of the resultant torque of both links is then given as the following:

$$\Gamma_1 = -\Gamma_2 = -K_{\mathrm{AP}} p \tag{5-153}$$

In general, the value of *dt* should not be too large where the links penetrate though the constraint region in one time step without meeting any of the rotary limit conditions.

## D.II Momentum-Balanced Method

The Momentum-balanced method uses conservation of momentum principle. The rotary momentum of both links before and after the limit boundary impact should be equal. The coefficient of restitution is involved and the following equations are used

$$\dot{\theta}_2' = \frac{\dot{\theta}_2(I_2 - I_1 e) + I_1\dot{\theta}_1(1+e)}{I_2 + I_1} \tag{5-154}$$

$$\dot{\theta}_1' = \dot{\theta}_2' - e(\dot{\theta}_1 - \dot{\theta}_2) \tag{5-155}$$

The post-contact rotary velocity of the links is denoted as $\dot{\theta}_1'$ and $\dot{\theta}_2'$. The coefficient of restitution is denoted as *e* in Equations (5-154) and (5-155). The displacement of the free node of respective link is updated accordingly.

## D.III Forced-Displacement Method

The Forced-displacement method has limited usage. The rotary displacement of the links is altered physically to cope with the rotary constraints. Changes in implementation are normally required from a problem to another. In addition, accurate post-contact response has to be known before hand by the user, in order to direct the response of the rotary displacement distribution. This method is not practical for solving complicated problems.

### D.IV Energy Conservation Method

Conservation of energy is used to predict the impact response of the rotating bodies at the limit boundary. Assumption is made that both the bodies will rotate at the same post-contact rotary velocity and no energy is lost. The method is achieved with the following equations.

$$\dot{\theta}^{\mathrm{II}} = \frac{\dfrac{I_1\dot{\theta}_1^2\dot{\theta}_1}{\left|\dot{\theta}_1\right|} + \dfrac{I_2\dot{\theta}_2^2\dot{\theta}_2}{\left|\dot{\theta}_2\right|}}{I_1 + I_2} \tag{5-156}$$

$$\dot{\theta}_1' = \dot{\theta}_2' = \sqrt{\left|\dot{\theta}^{\mathrm{II}}\right|}\frac{\dot{\theta}^{\mathrm{II}}}{\left|\dot{\theta}^{\mathrm{II}}\right|} \tag{5-157}$$

where $\dot{\theta}^{\mathrm{II}}$ is a signed square of the predicted rotary velocities. It is a transitional value. The displacement of the link elements' free node is updated accordingly with the above equations.

## E. Locking

Locking can be activated, where no further rotary motion is allowed after reaching the rotary limits. Thus, recovery is impossible for the rest of the simulation motion. A user prescribed Boolean, *bLock*, is used. When locking, one of the above mentioned methods can be used for both directions (forward and reverse) and no releasing is possible. Vibration is expected accompanying the locking mechanism.

## F. Viscous Damping

Rotary viscous damping is included in the element. Damping coefficient, *C*, is used. The viscous damping torque is defined as the following,

$$\Gamma_{C,1} = -\Gamma_{C,2} = C\left(\dot{\theta}_2 - \dot{\theta}_1\right) \tag{5-158}$$

## G. Stability of the Revolute Element

Stability is an important issue in computational mechanics. The Stability of the Central Difference temporal integration scheme is given as the following according to Courant-Friedrichs-Levy criterion [102]:

$$\Delta t_{\text{critical}} = \frac{L}{C} \tag{5-159}$$

where $L$ is the minimum link length of the 1-D element and $C$ is the maximum wave propagation velocity. Based on the fact, a new criterion is suggested by the authors when involving the revolute element.

$$\Delta t_{\text{critical}} = \min\left(\frac{L}{C}, \sqrt{\frac{\rho I L}{K}}\right) \tag{5-160}$$

where $\rho$, $I$, and $K$ are the minimum density, minimum second moment of area around the neutral axis and the rotary spring stiffness involved of both connected bodies. The development of the criterion is inspired with the relation among bending moment, $\Gamma$, Young's modulus, $E$, and bending angle, $\theta$.

$$\Gamma = \int_{-h/2}^{h/2} \frac{E\theta b}{L} r^2 dr \tag{5-161}$$

where $b$, $h$ and $L$ are the width, height and length of the link element respectively. The Adaptive Stiffness value should be revised with the following relation when involving locking.

$$K \leq \frac{0.81\rho I L}{dt^2} \tag{5-162}$$

Instability may be observed if the criterion is not followed with the revolute element especially involving very long contacting time and locking mechanism. Further investigation has to be carried out to have the full coverage of the revolute element stability with central difference temporal integration scheme. The criterion can be served as a preliminary checking.

### 5.11.3 CHARACTERIZATIONS OF ROTARY LIMIT RESPONSE METHODS

Comparisons are made among the developed adaptive rotary stiffness method, the conventional Penalty-like method, momentum-balanced method, forced-displacement method and energy conservation method. Formulation of the methods (momentum-balanced, force-displacement and energy) will be explained briefly in this section. The characteristics of the models are compared while solving a numerical problem.

To study the characteristics of the revolute element, a simple problem was solved with different methods and releasing methods. Figure 5-68 shows the schematic diagram of the problem. Two links are connected with a pivoted revolute joint with initial rotary velocity of $\dot{\theta}_0$. $L$ is the link length and $\theta_0$ is the initial rotary displacement of the right link to the left link. The revolute joint reverse limit is denoted as $\theta_r$. The cross-section of the link is denoted as $H$ and $W$. Each link is discretised into 10 elements. The material of the links is perfect elastic with density and stiffness of aluminium, 70 GN/m$^2$ and 2710 kg/m$^3$ respectively.

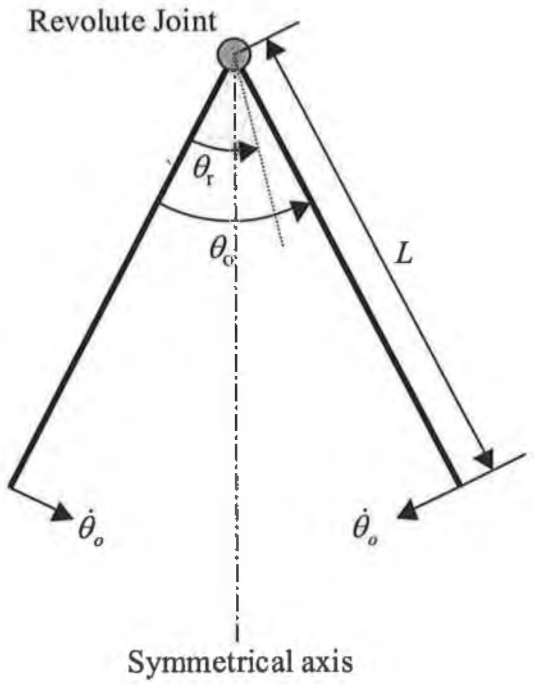


Figure 5-68: Schematic diagram of the simple revolute joint problem

Table 5-10: Parametric values of characterisation example

| Parameter | Value |
|---|---|
| Link length, $L$ | 1 m |
| Height, $H$ | 0.2 m |
| Width, $W$ | 0.05 m |
| Density, $\rho$ | 2710 kg/m$^3$ |
| Young's modulus, $E$ | 70 G Pa |
| Initial rotary displacement, $\theta_o$ | 180° |
| Reverse rotary limit, $\theta_r$ | 176° |
| Time step size, $dt$ | 15e-6 sec |

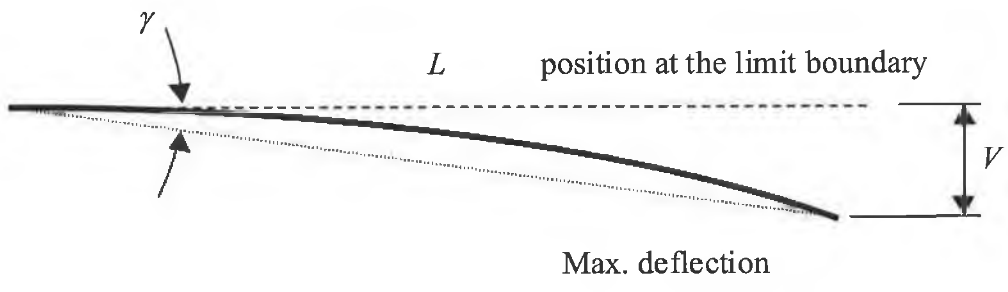


Figure 5-69: Deflection after reaching rotary limit

Table 5-10 shows the numerical values used for both structural and revolute modelling. An approximated analytical time-response relation is derived. Figure 5-69 shows the schematic diagram of the proposed analytical model from the time a link starts contacting the limit boundary till its maximum deformation. The $L$ and $V$ are the link length and maximum deflection respectively. The maximum deflection is given by the following formula.

$$V = \frac{FL^3}{3EI} \tag{5-163}$$

where $F$ is the force applied at the tip of the link to cause the deflection of $V$. the term $I$ is the second moment of area around neutral axis. The model is approximated with the assumption that the deflection $V$ is very small compared to the link length $L$ (where $\gamma \approx V/L$). The approximated natural frequency of the system is given as below:

$$\omega_n = \frac{H}{2L}\sqrt{\frac{3E}{(L^2 + H^2)\rho}} \tag{5-164}$$

With Equations (5-159) and (5-160), the following rotational displacement response of nodes K and L (the preceding and succeeding nodes of the joint node) at the connected joint of the link can then be developed.

$$\theta = \begin{cases} \dot{\theta}_o t, & t < t_L \\ \dot{\theta}_o t_L, & t_L \le t \le t_R \\ \dot{\theta}_o t_L - \dot{\theta}_o (t - t_R), & t > t_R \end{cases} \tag{5-165}$$

$$t_L = \frac{(\theta_o - \theta_r)}{\dot{\theta}_o} \tag{5-166}$$

$$t_R = t_L + \frac{4\pi L}{H}\sqrt{\frac{3E}{(L^2 + H^2)\rho}} \tag{5-167}$$

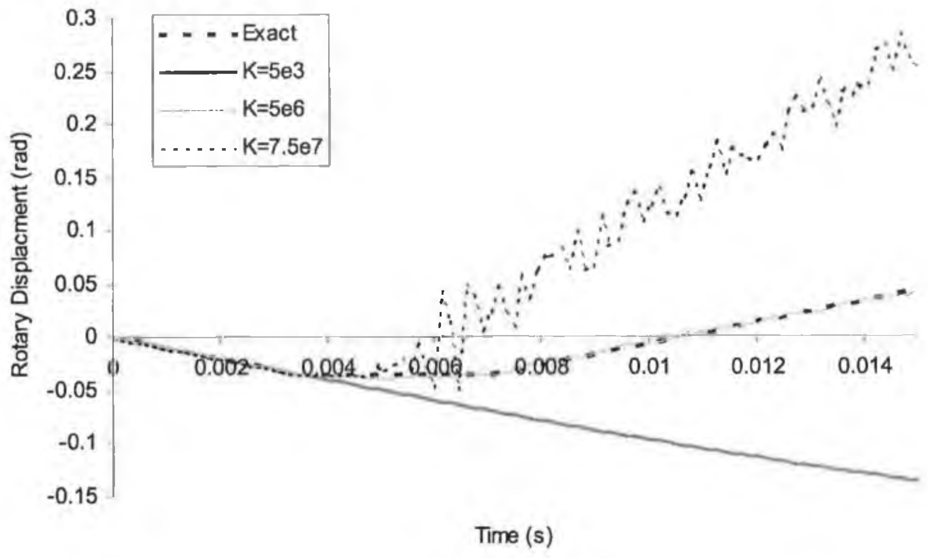


Figure 5-70: Rotary displacement history with ANSYS COMBIN7 (Penalty-like method)

Characteristics of different user-defined stiffness with COMBIN7 Penalty-like method are shown in Figure 5-70. Releasing conditions of Equation (5-144) (RC13) are used. Refined stiffness value of 5x10$^{6}$ Nm/rad shows a great compromise with the exact solution. The refinement is done through trial and error. A low stiffness value will cause an unacceptable significant rotary penetration and may even penetrate though the entire limit range. The graph also shows the over-response with high stiffness value. For a simple problem, trial and error method can be used to get a proper stiffness value, which yields good results. One is hardly able to get a refined stiffness value through trial and error when handling a complicated problem. This is particularly true when one cannot tell the proper response of the system before hand. Responses with different releasing conditions with Penalty-like method are shown in Figure 5-71. Releasing conditions of Equation (5-143) (RC12) shows more sensitive response compare to RC13 with the same stiffness response. A lower stiffness value can be used to get a satisfactory result, which is shown.

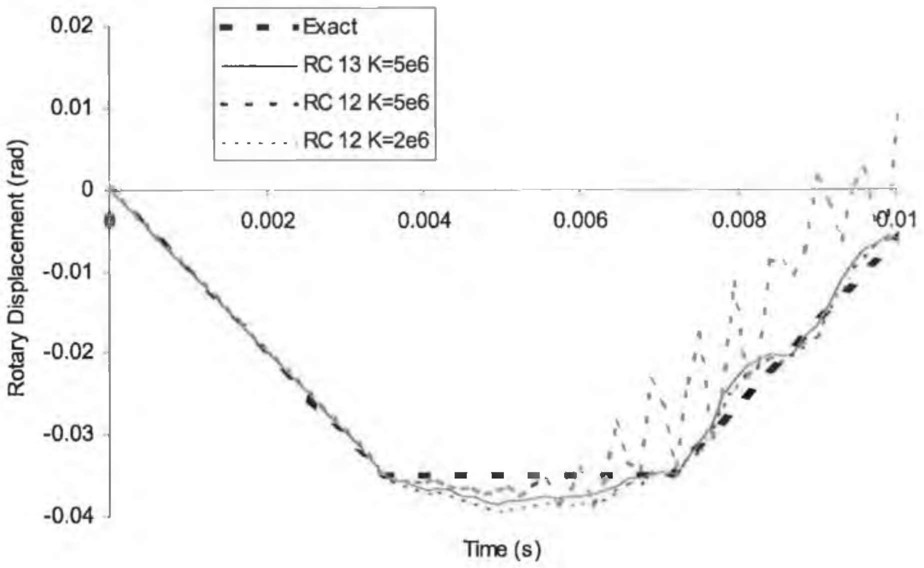


Figure 5-71: Penalty-like method with different releasing conditions

The Adaptive Stiffness method shows good correlation to the exact solution in Figure 5-72. Both the Adaptive and the fine-tuned Penalty-like methods can be considered giving the same response. A maximum penetration of 0.0045 rad can be observed in both methods with the same releasing condition of Equation (5-144). The Releasing conditions of Equation (5-143) are not suitable although they give less penetration magnitude. They give extra excitation towards the whole system, which can be observed in Figure 5-72. The Adaptive Stiffness method with RC13 predicts an excellent stiffness value.

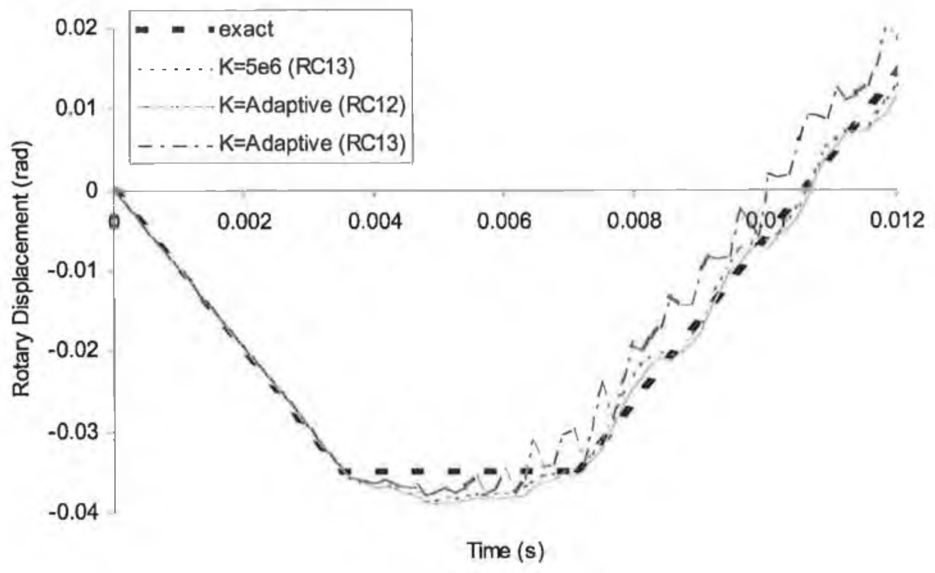


Figure 5-72: Rotary displacement history with Adaptive Stiffness method

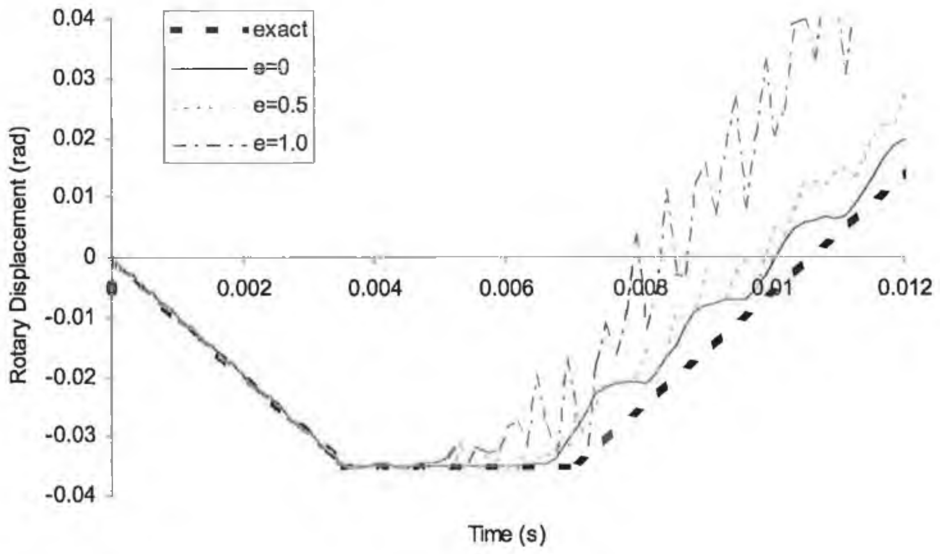


Figure 5-73: Rotary displacement history with Momentum-balanced method (e = coefficient of restitution)

Figure 5-73 shows the response of the Momentum-balanced method. Only releasing conditions of Equation (5-144) are applicable for this method. With studies on three different coefficients of restitution, only the perfect plastic model is suitable. The other two create extra excitation especially with the perfect elastic Momentum-balanced method. The method with zero coefficient of restitution is good enough to be used. A good response is predicted with zero coefficient of restitution, especially no significant penetration occurs. The post-impact response is not as satisfactory as the Adaptive stiffness method.

Corresponding to the Forced-displacement method and the Energy Conservation method, Figure 5-74 is plotted. Both methods show pre-mature excitation compared to the exact solution. The excitation produced by the Forced-displacement method is higher than by the Energy Conservation method. From this context, the Forced-displacement method is not always the best/safest method (not in terms of flexibility) to solve a constraint problem.

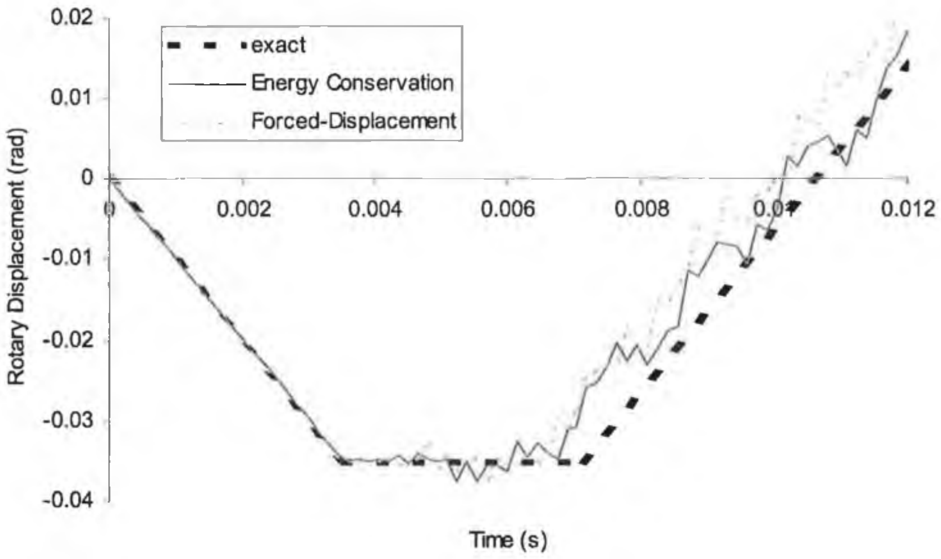


Figure 5-74: Rotary displacement history with Energy Conservation method and Forced-displacement method

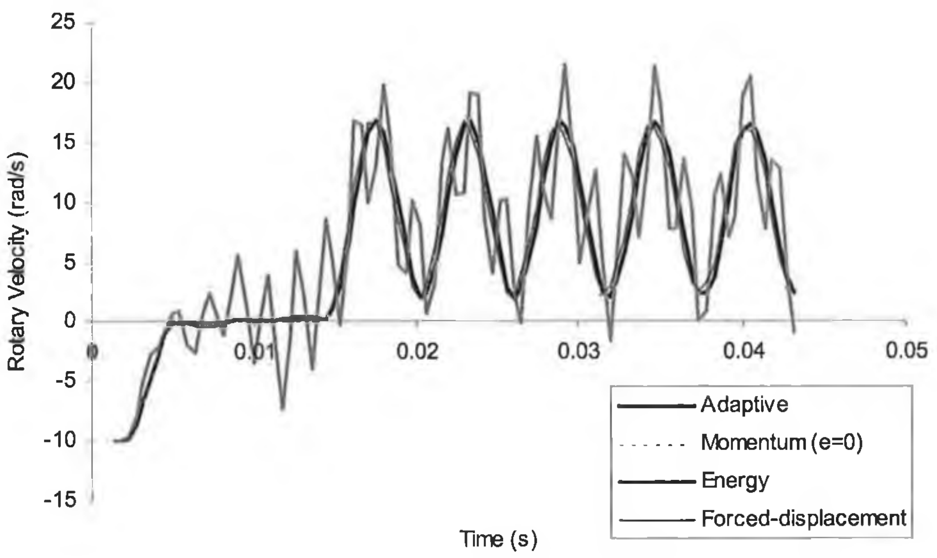


Figure 5-75: Rotary velocity profiles of different method in solving the simple revolute joint problem with thin thickness (long contact time)

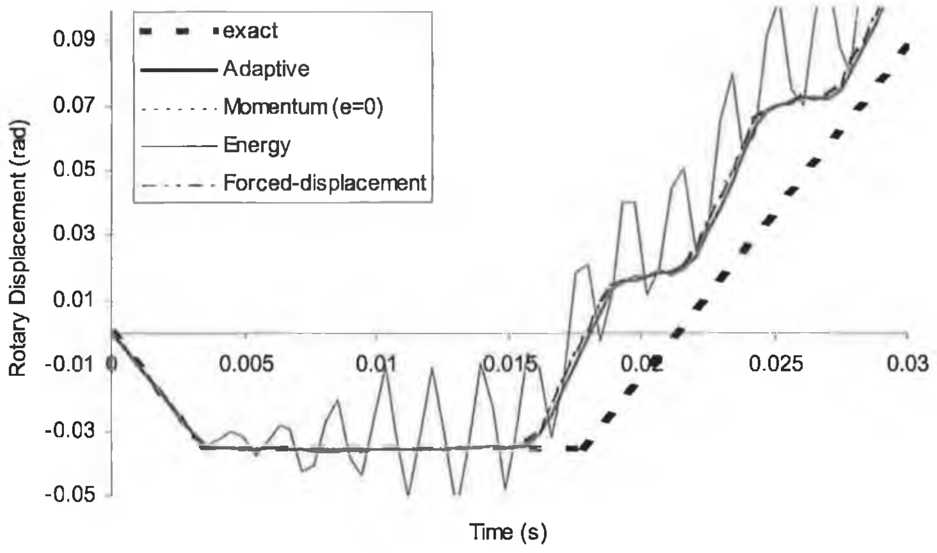


Figure 5-76: Rotary displacement profile of different method in solving the simple revolute joint problem with thin thickness (long contact time)

To study the effect on longer contact time, the value of *H* in the problem is decreased down to 0.05m, which results a longer contacting time. Different rotary limit models are used to solve the problem. Figure 5-75 shows the rotary velocity profile of the preceding and succeeding nodes of the joint node. The post-contact sinusoidal response is due to internal vibration of the connected structures (links). It is clear that the Energy Conservation method excites the system and is not suitable to be used comparing to others. The Adaptive Stiffness, Momentum-balanced and Forced-displacement methods predicts similar responses. No significant change is evident in velocity profiles during contact with these methods. Again, the rotary displacements with different methods are shown in Figure 5-76. Deviation from the exact solution is expected as assumption is made above in developing the exact formulation. All but the energy method give good response.

For a more complicated problem, such as the one shown in Figure 5-77, the Forced-displacement method proved to be inapplicable. The displacement responses of both the links are not known. Investigation of the revolute joint at particular arrangement has to be made in order to make proper prediction of displacement adjustments. The COMBIN7 Penalty-like, Adaptive Stiffness, Momentum-balanced and Energy conservation methods are capable of handling such situations. Considering a complex situation where several revolute joints with rotary limits connecting several different links (huge variance in dimension and material properties from link to link), the Momentum-balanced method and the Energy-conservation method creates instability. This is possibly due to the disturbance the physical displacement created when applying the method. Furthermore, several revolute joints which are placed nearby superimpose the effect of the disturbances towards the systems and thus instability is observed.

As a whole, the Energy-conservation method is not a good method and should not be used. The Forced-displacement method is excellent to be used when longer contact time involved. It predicts a proper response during contact. The method's shortcomings are lack of flexibility and difficult in generalisation. In addition, Forced-displacement method does not provide a good response to the system when solving short contacting time problems. The Momentum-balanced method with zero coefficient of restitution is suitable to be used, in solving both short and long contact time problems. Instability may be reported in solving a complex revolute joints problem. Thus careful monitoring is required when using the method. Last but not least, the Adaptive Stiffness method is able to predict a good rotary stiffness. Similar to the COMBIN7 Penalty-like method, it is suitable to be used in all situations.

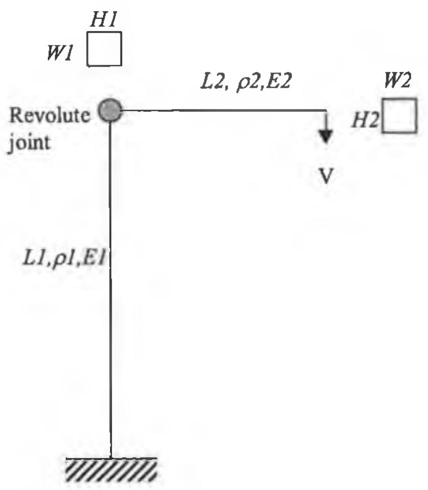


Figure 5-77: More complicated example

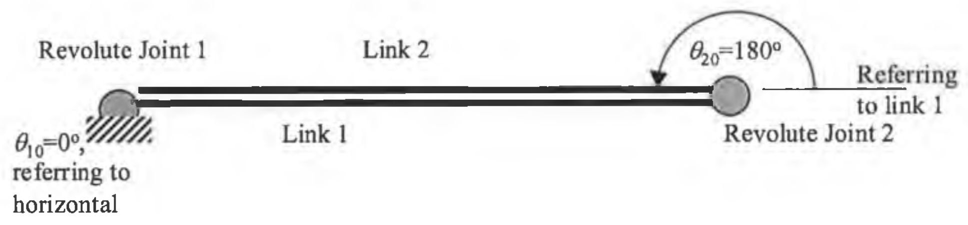


Figure 5-78: Schematic diagram of the two-link undergoing locking problem

### 5.11.4 NUMERICAL EXAMPLE

Space structures have large dimensions low mass to size ratio, large inertia and relative low structure rigidity. These structures are originally in stowed configurations and are deployed into their full size in space. At the final stage of the deployment, locking at the joints is required. Vibrations are expected especially after the locking. Nagaraj *et al.* [147] have constructed a mathematical model and an experimental rig to study the dynamics of a two-link flexible system undergoing locking. Schematic representation of the system is shown in Figure 5-78. A torsion spring has been place at both joints. Physical parameters of the system given by the literature are shown in Table 5-11. Figure 5-79 and Figure 5-80 show the experimental set-up of Nagaraj *et al.* [147].

The new revolute element is used to simulate the problem. Both links are discretised into 10 Node-Layer link elements. Perfect elastic material model is used. Because of the locking mechanism and the nature of problem, which involved large vibration, the value of *K* is selected according to the stability criterion mention previously for joint 2. Two trials are run with different time step size and value of *K*. They are 15 microseconds with 150 Nm/rad and 7.5 microseconds with 600 Nm/rad. Forced-displacement method is used for joint 1.

Table 5-11: Physical Parameters of Numerical Example [147]

| Link1 and Joint 1 | |
|---|---|
| Length (m) | 1.006423 |
| Cross-sectional area (m$^2$) | 1.78076e-4 |
| Thickness (m) | 4.4519e-3 |
| Area moment of inertia (m$^4$) | 2.94113e-10 |
| Rotary inertia (kgm$^2$) | 8.5948e-4 |
| Flexural rigidity | 20.5879 |
| Young's Modulus | 7e10 |
| Link mass (kg) | 0.52334 |
| Tip Mass (kg) | 1.2 |
| Torque (Nm) | 0.03825 |
| Torsion spring stiffness (Nm/rad) | 0.0789 |
| Pre-rotation angle (°) | 300 |
| Initial rotary displacement (°) | 0 |
| Locking rotary displacement (°) | 90 |
| **Link 2 and Joint 2** | |
| Length (m) | 0.9945 |
| Cross-sectional area (m$^2$) | 1.7748e-4 |
| Thickness (m) | 4.437e-3 |
| Area moment of inertia (m$^4$) | 2.9117e-10 |
| Rotary inertia (kgm$^2$) | 9.256e-4 |
| Flexural rigidity | 20.3819 |
| Young's Modulus | 7e10 |
| Link mass (kg) | 0.42958 |
| Tip Mass (kg) | 0.336 |
| Torque (Nm) | 0.0225 |
| Torsion spring stiffness (Nm/rad) | 0.0768 |
| Pre-rotation angle (°) | -60 |
| Initial rotary displacement (°) | 180 |
| Locking rotary displacement (°) | 0 |

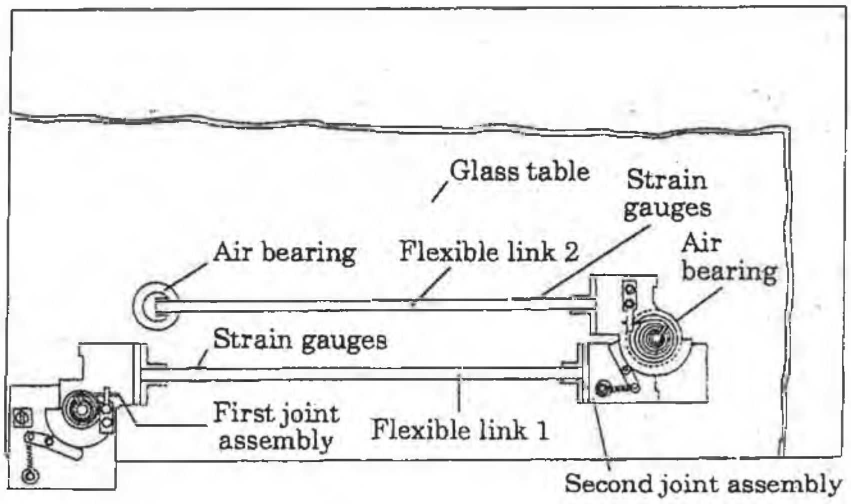


Figure 5-79: Experimental set-up of the two-link undergoing locking problem [147]

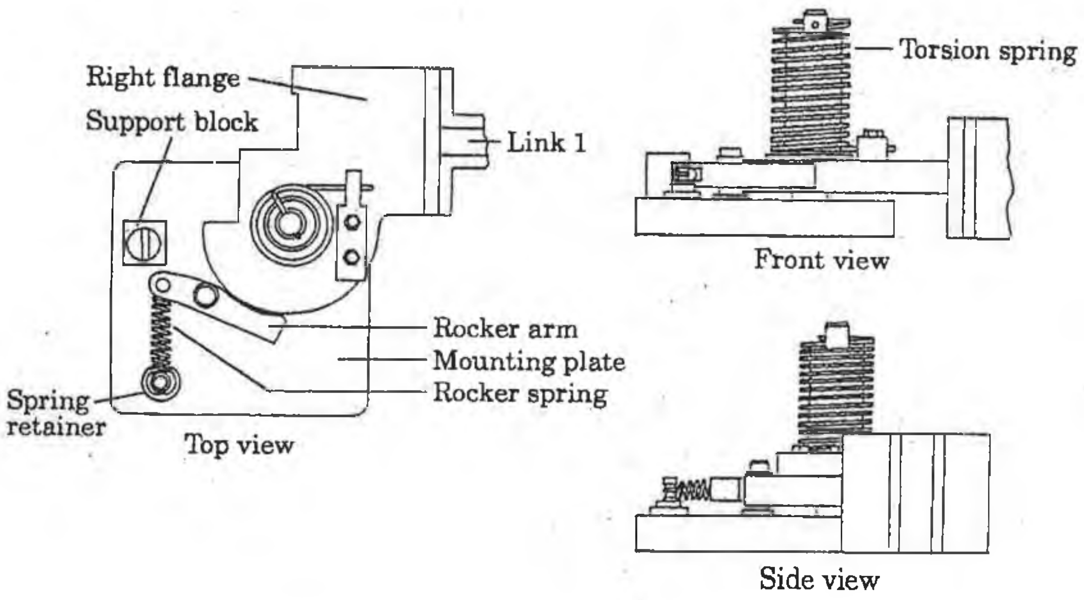


Figure 5-80: First joint assembly initial configuration [147]

Figures 5-81 and 5-82 show the rotary displacement histories at both joints. Experimental results, exact solution (rigid model) and Nagaraj *et al.*'s model is compared. It is obvious from Figure 5-81 that Nagaraj *et al.*'s model and the models developed in this study complement each other before the locking at joint 2 occurred. After the locking, present models tend to follow the path of rigid model, whereas the Nagaraj *et al.*'s model deviates from the rigid model and follows the experimental result. Additional consideration was applied by Nagaraj *et al.* where a part of the kinetic energy of link 2 is transformed into flexural strain energy and flexural kinetic energy. This caused high peak tip displacement up to 20.5% of the total length after locking [147]. As perfect elastic material model used in the present solutions, the responses are expected to follow rigid model. The use of smaller time step size with higher rotary stiffness produced a slightly faster recovery from the locking. The difference is small and the peak of the vibration profile is very close to the smaller time step size model. Figure 5-82 shows that the present models have a closer response to the experimental result compare to Nagaraj *et al.*'s. After the locking, smaller time step size with higher rotary stiffness give more stable response in term of rigidity of the locking joint 2. Small amplitude of oscillation is observed with lower rotary stiffness model. Please note that the Nagaraj *et al.*'s model stopped predicting after reaching the rotary limit (approximate at time = 3 second).

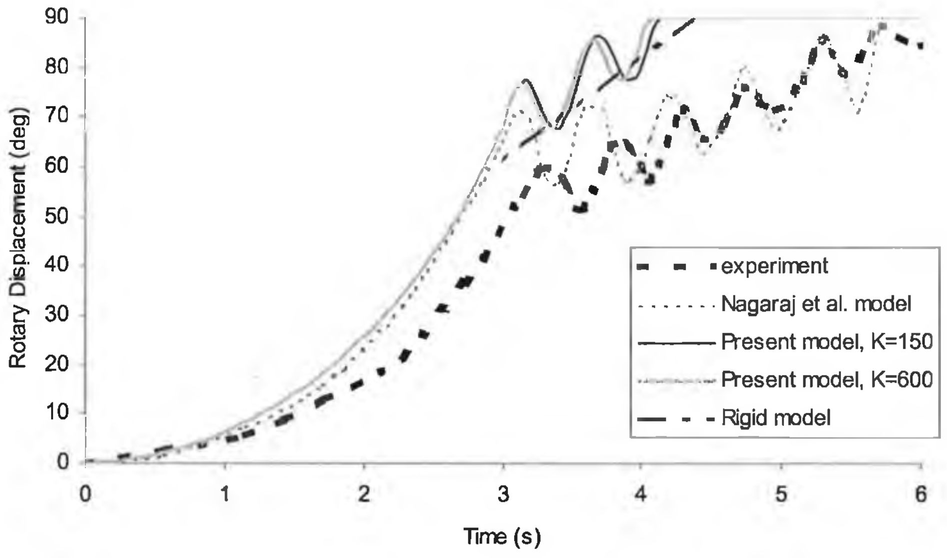


Figure 5-81: Rotary displacement profile at joint 1 of the tow-link undergoing locking problem

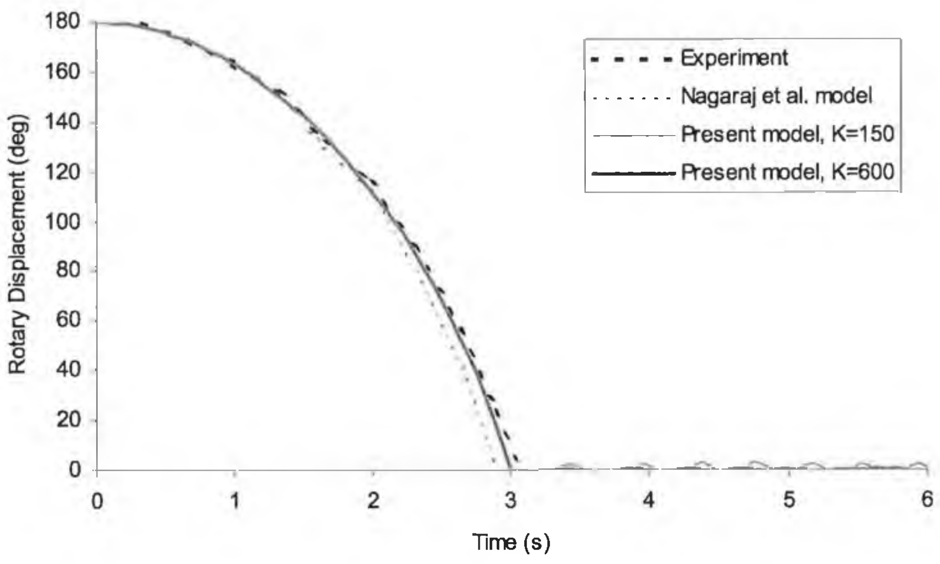


Figure 5-82: Rotary displacement profile at joint 2 of the two-link undergoing locking problem

### 5.11.5 DISCUSSIONS AND SUMMARIES

Development of an explicit revolute joint element is explained in details in this section. The element is able to handle revolute joint with rotary limits, locking, resistive moment and damping effects. The effects of different contact and releasing conditions are shown. The element structure is presented and the element formulations are described. Several possible ways of handling the rotary limits are shown. Two new formulations developed in this work to tackle the rotary limits are shown. They are Adaptive Stiffness method and Energy method. New implementation of the Momentum-balanced method is shown. The conventional COMBIN7 Penalty-like method and the forced-displacement method are included and explained. The differences among their characteristics are discussed while solving a numerical problem. Several drawbacks and limitations of the methods are discussed briefly. A new stability criterion is suggested when using revolute joint, especially involving locking mechanism. Further study in the computational stability of the revolute joint is required.

The Energy method is generally not a good method although it can hold the rotating links from collapsing. The conventional Forced-displacement is good but suffering from flexibility and showing inefficiency in solving short contact time problem. Penalty-like method needs user-defined parameters, which is normally unknown and not important to the user. The methods developed in this study showed that the capability of the new Adaptive Stiffness method to predict a proper rotary stiffness to handle rotary limits, are better compared to the user defined value in Penalty-like method. The effects of too high and too low of the stiffness value are shown. It is also shown that the use of the Momentum-balanced method with coefficient of restitution ranging from perfect plastic to perfect elastic is feasible but the perfect plastic Momentum-balanced method is preferred. The Adaptive Stiffness method with releasing conditions of Equation (5-144) predicted excellent response when compared to the exact solution.

An example from the literature, which involved locking mechanism, is solved with the Node-Layer link element and the new revolute element. The stiffness value is predicted from the stability criterion. The numerical results show very good correlation to the exact solution. When comparing to the experimental result, a reasonable prediction is given as perfect elastic material model is used.

The new revolute joint element can be incorporated into any implicit, explicit, finite difference or finite element solver. In addition, the new Adaptive Stiffness method with releasing conditions of Equation (5-144) can be used as a replacement to the conventional Penalty-like method. When, involving locking mechanism, the stability criterion should be used to prevent instability and the stiffness value has to be revised.

# Chapter 6: HUMAN MOTION SIMULATION

## 6.1 INTRODUCTION

In this chapter the use of the MSS system as a simulation tool to model human dynamic motion subjected to frontal collision is explored. Particular interests have been given to the crash of a motorcycle with a human rider. The chapter begins with illustration of the development of a human model. Information gathered from the literatures is incorporated into the development in order to create a realistic model. Preliminary studies on human motion, particularly the influence of the acceleration/deceleration of the head during the collision, have been carried out with the developed human model. A $2^k$ factorial design of experiment method is used to minimise the simulations required to determine the most significant effect(s) towards the head injuries. Head Injury Criterion (HIC) is used as the measure. Then a motorcycle model is developed. The development details are exhibited. The motorcycle with the human model is developed and simulated under the MSS. In the later part of the chapter, conceptual design of a sled test rig is illustrated.

## 6.2 HUMAN MODEL

A 10-segment idealised human model has been developed. The segments are 2-feet, 2-shanks, 2-thighs, 2-hands, 2-forearms, 2-upper arms, abdomen and pelvic, thorax, neck, and head. In geometrical construction of the model, each segment is represented by a line. Line constraints are applied in geometrical construction of the human model. This makes the change of the human model parameters, such as posture and segment-length, easier. All the line constraints let the first node of the element as the fixed position node. A typical human model is shown as Figure 6-1.

### 6.2.1 SEGMENT LENGTHS

Drillis and Contini [161]'s human body segment parameter is used to define the human body segment lengths relative to overall body height. The following table shows the ratio used in defining the segment-lengths.

Table 6-1: Body segment lengths expressed as a fraction of body height

| Segment | Ratio to overall height |
|---|---|
| Foot Length | 0.152 |
| Foot Height | 0.039 |
| Shank | 0.246 |
| Thigh | 0.245 |
| Abdomen & Pelvic | 0.145 |
| Thorax | 0.143 |
| Neck | 0.052 |
| Head | 0.13 |
| Upper arm | 0.186 |
| Forearm | 0.146 |
| Hand | 0.108 |

Table 6-2: Body segment lengths expressed as a fraction of body height

| Segment | Ratio to overall weight |
|---|---|
| 2 Feet | 0.03 |
| 2 Shanks | 0.086 |
| 2 Thighs | 0.206 |
| Abdomen & Pelvic | 0.294 |
| Thorax | 0.213 |
| Neck | 0.009 |
| Head | 0.064 |
| 2 Upper arms | 0.052 |
| 2 Forearms | 0.032 |
| 2 Hands | 0.014 |

### 6.2.2 SEGMENT WEIGHTS

Cadaver segment weights of Clauser *et al.*'s study [159] are used in modelling the human segment-weights. Assumptions on certain human segment-weights have been made, as no data is available from the literature. The following table is weight ratio used with respect to the whole body.

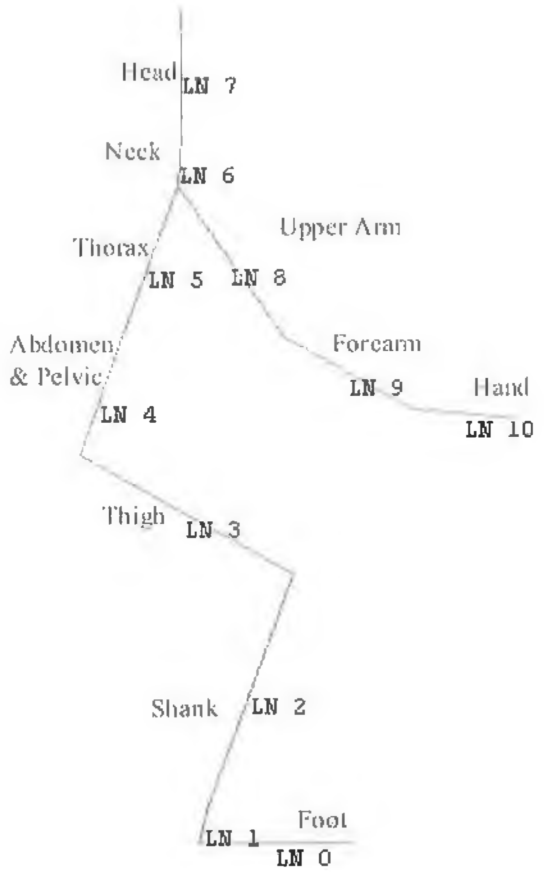


Figure 6-1: Human Geometry model

### 6.2.3 MATERIAL CHARACTERISTICS

Bone property is used in modelling the segment material. The bone property of reference [227] is used, where the bone density is given between 1600kg/m$^3$ to 1700kg/m$^3$. The Young's modulus is found to be 1.74 GPa and the ultimate tensile strength is 135 MPa. The elongation at fracture is found to be around three to four percent. No further detailed information is available, thus the above information is used to define the material characteristics for the human model. Figure 6-2 shows the idealised stress-strain relationships of the model. Strain-rate effect is ignored, as no information is available. The imprecise information on the bone property and lack of information on the bone-flesh-skin relationships, bone fractures and soft tissue injuries are beyond the study of this section. These studies are possible with present simulation system provided having proper and adequate information. New material model may be required, which can easily be incorporated with the OOP design of the MSS. The MSS processor input data file's material definition will look like the following:

```
Mat 0, 1700, 8,3, 1000, 0.001, 17.4e+009,1e+008, 5e+009,1.15e+008,
1.00e+009,1.35e+008;
```

Coding 6-1: Bone material definition in MSS input data file

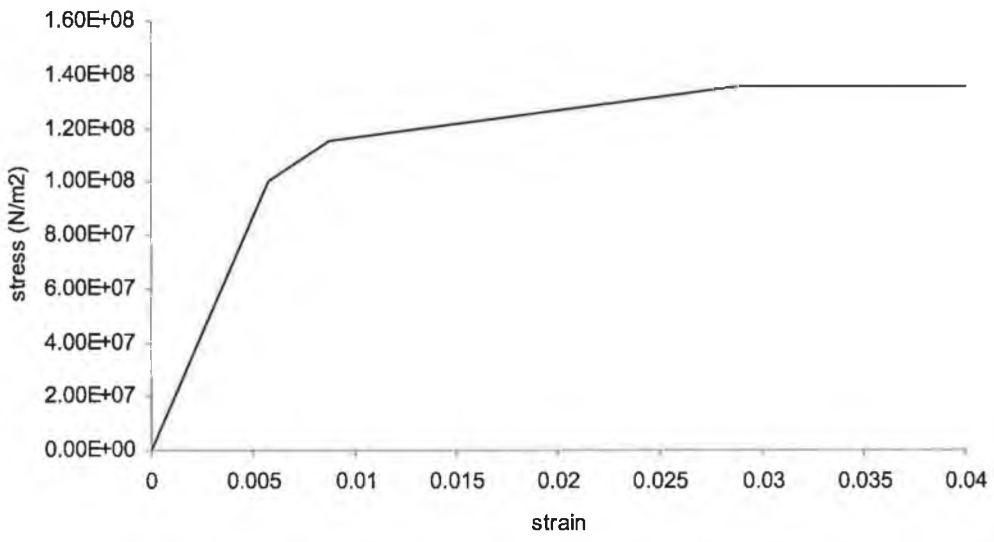


Figure 6-2: Idealised stress-strain relation of bone

### 6.2.4 SEGMENT CROSS SECTIONS

A segment's density, length, and weight are fixed with the values from the literature. Thus the idealised cross section has to be adjusted to compensate the above mentioned properties. In order to avoid the complexity of the bending stiffness, the cross-sectional thickness of all the elements are fixed with a constant value, which is 0.07m and the width is varied accordingly. MSS pre-processor provides a tool to automate the arrangement (*Assign Material*).

### 6.2.5 DISCRETISATION OF THE HUMAN MODEL

The human model is discretised into finite number of elements. The number of nodes of each segment is shown in Table 6-3. Figure 6-3 shows the corresponding node numbers after node-link generation with coincide-nodes eliminated.

### 6.2.6 HUMAN JOINTS

The human model consists of 12 revolute joints, which includes the vertebral rotational joints. They are ankle, knee, hip, wrist, elbow, shoulder and six vertebral joints. The vertebral movements are limited to 5 joints, which representing L4-S1, L3-L4, L1-L3, C5-T1, C3-C5, and C1-C3 of the spine. The corresponding node reference number and link reference numbers of the joints are shown as the following table.

Table 6-3: Body segments discretisation

| Segment | Number of nodes |
|---|---|
| 2 Feet | 4 |
| 2 Foot Heights | 2 |
| 2 Shanks | 6 |
| 2 Thighs | 6 |
| Abdomen & Pelvic | 4 |
| Thorax | 4 |
| Neck | 3 |
| Head | 3 |
| 2 Upper arms | 5 |
| 2 Forearms | 4 |
| 2 Hands | 4 |

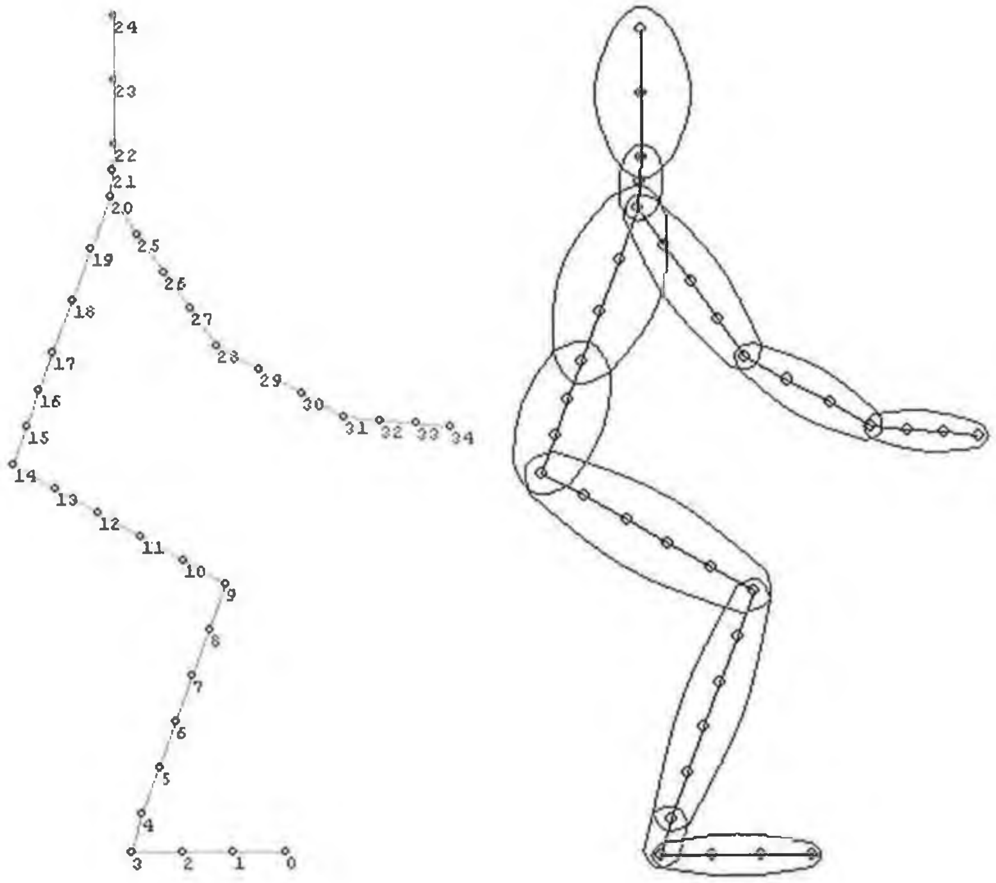


Figure 6-3: Discretised human model with displaying node numbers

Table 6-4: Corresponding node numbers of respective joints

| Joint | Link numbers | Node number | Reference Link |
|---|---|---|---|
| Ankle | 3, 4 | 4 | 3 |
| Knee | 8, 9 | 9 | 8 |
| hip | 13, 14 | 14 | 13 |
| Wrist | 30, 31 | 31 | 30 |
| Elbow | 27, 28 | 28 | 27 |
| Shoulder | 19, 24 | 20 | 19 |
| L4-S1 | 14, 15 | 15 | 14 |
| L3-L4 | 15, 16 | 16 | 15 |
| L1-L3 | 16, 17 | 17 | 16 |
| C5-T1 | 19, 20 | 20 | 19 |
| C3-C5 | 20, 21 | 21 | 20 |
| C1-C3 | 21, 22 | 22 | 21 |

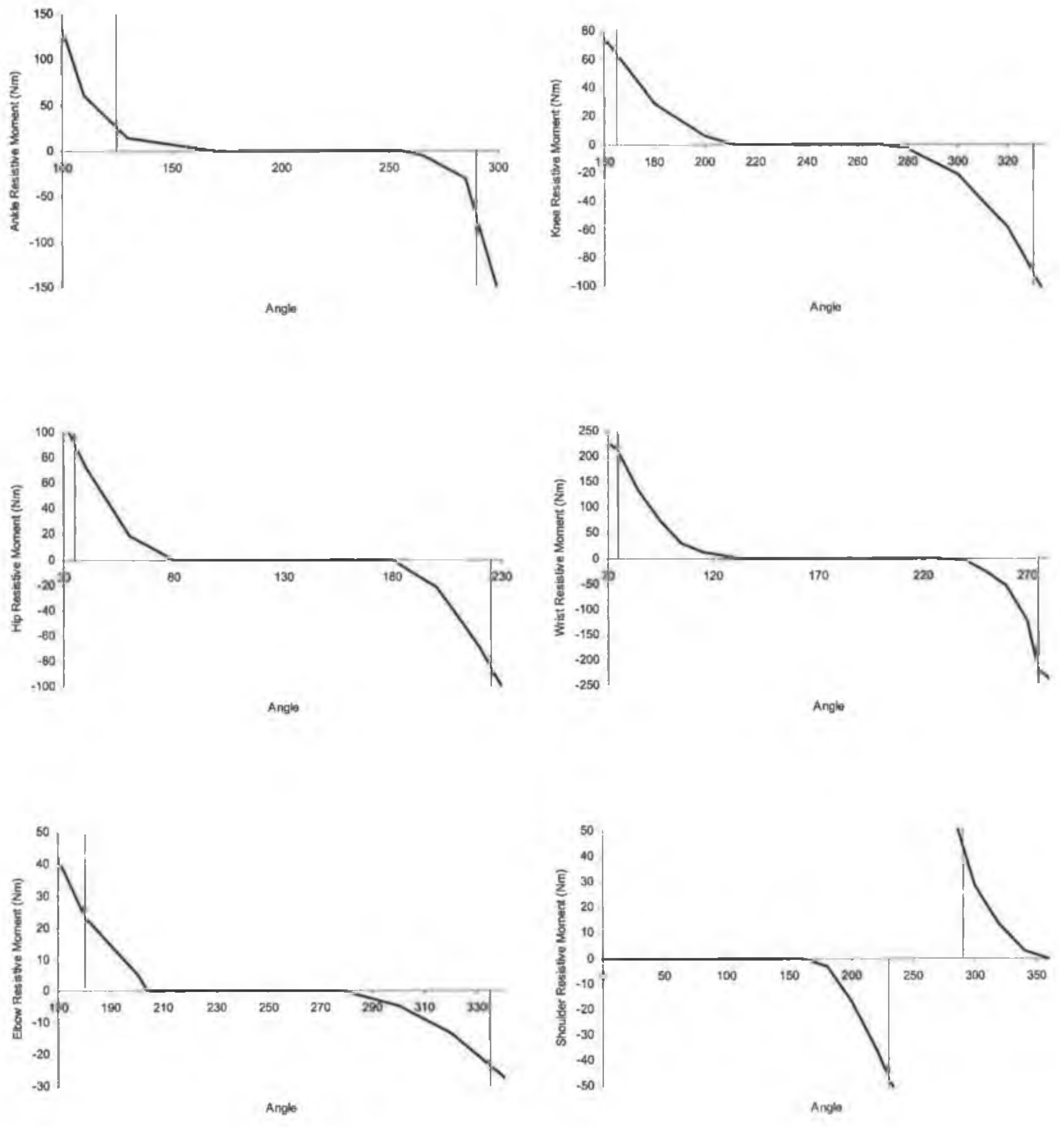


Figure 6-4: Resistive moment and rotary limits of ankle, knee, hip, wrist, elbow, and shoulder

### 6.2.7 JOINT RESISTIVE MOMENTS, ROTATION LIMITS AND DAMPING COEFFICIENTS

The joints are modelled as revolute joints. Ma *et al.*'s resistive torques model [169] is incorporated. The model is explained in section 2.10.1. The data are further interpreted and incorporated into the human model with revolute joint elements. Ma *et al.*'s model does not include the vertebral, thus additional manipulations were made in this study to model the joints. Data from Table 2-1 of Chapter 2 is used to define the stopping angles (rotation limits) of respective vertebral joint. Figures 6-4 and 6-5 show respective resistive moment versus angle chart. Angle is measured from the reference link in degrees. Vertical lines in the graphs are the rotary limits of respective joints. Damping coefficient of ankle, knee, hip, elbow, and wrist joints are set to 1.942 Nms/rad; the damping coefficient for other joints is 3.884 Nms/rad.

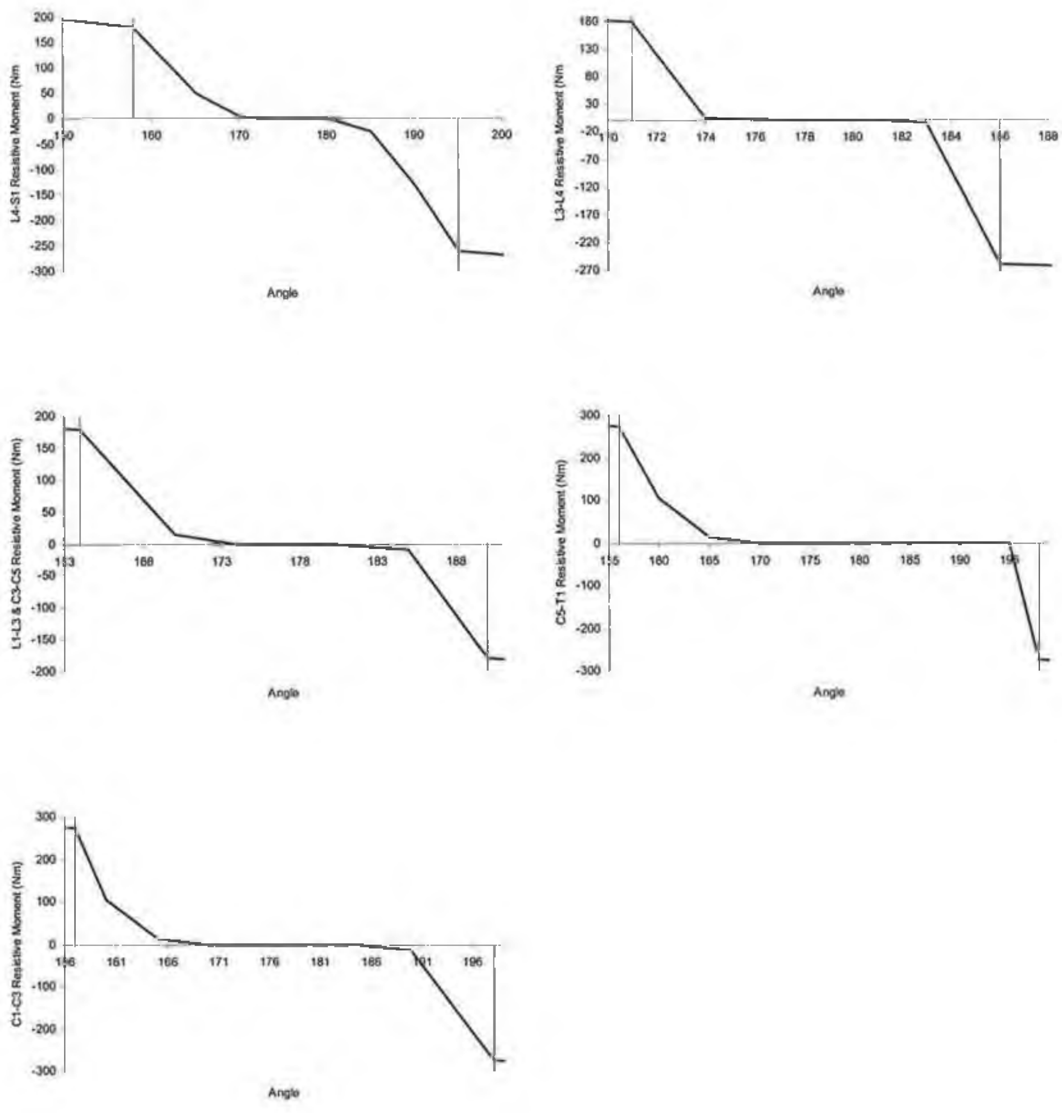


Figure 6-5: Resistive moment and rotary limits of L4-S1, L3-L4, L1-L3 & C3-C5, C5-T1, and C1C3

The following coding shows the definition of the revolute joint in MSS processor input data file format.

```
RMmt 0, 3,4, 1.942, 10, 290,360, 0,125, 0,500, 90,210, 110,60, 130,15,
170,0, 255,0, 265,-5, 285,-30, 305,-200, 360,-500;
RMmt 1, 13,14, 1.942, 15, 225,360, 0,35, 0,241.077, 20,145.84, 40,72.054,
60,18.4597, 80,0, 180,0, 200,-22.1406, 220,-69.6892, 240,-132.918, 260,-
209.612, 280,-315.598, 300,-510.902, 320,-582.794, 340,362.531, 360,241.077;
RMmt 2, 14,15, 3.884, 10, 195,360, 0,158, 0,500, 158,180, 165,50, 170,5,
172,0, 180,0, 185,-25, 190,-130, 195,-260, 360,-500;
RMmt 3, 15,16, 3.884, 9, 186,360, 0,171, 0,500, 171,180, 174,5, 177,0,
180,0, 181,0, 183,-5, 186,-260, 360,-500;
RMmt 4, 16,17, 3.884, 8, 190,360, 0,164, 0,500, 164,180, 170,15, 174,0,
180,0, 185,-10, 190,-180, 360,-500;
RMmt 5, 20,21, 3.884, 9, 190,360, 0,164, 0,500, 164,180, 170,15, 174,0,
180,0, 181,0, 185,-10, 190,-180, 360,-500;
RMmt 6, 21,22, 3.884, 9, 198,360, 0,157, 0,500, 157,275, 160,105, 165,15,
170,0, 185,0, 190,-15, 198,-275, 360,-500;
RMmt 7, 19,20, 3.884, 9, 198,360, 0,156, 0,500, 156,275, 160,105, 165,15,
170,0, 185,0, 195,0, 198,-275, 360,-500;
RMmt 8, 30,31, 1.942, 15, 275,360, 0,75, 0,500, 75,210, 85,130, 95,75,
105,30, 115,12, 125,5, 130,0, 225,0, 240,-5, 250,-26, 260,-55, 270,-125,
275,-220, 360,-500;
RMmt 9, 27,28, 1.942, 19, 335,360, 0,180, 0,-45.5119, 20,-70.1469, 40,-
101.205, 60,-139.239, 80,517.715, 100,343.951, 120,213.938, 140,121.137,
160,59.9279, 180,23.3141, 200,5.21669, 204,0, 215,0, 279,0, 280,-0.123576,
300,-4.60543, 320,-13.2949, 340,-26.7458, 360,-45.5119;
RMmt 10, 8,9, 1.942, 18, 330,360, 0,165, 0,-214.877, 20,-343.292, 40,-
512.065, 60,-726.277, 80,631.132, 100,421.759, 120,264.087, 140,150.893,
160,74.9507, 180,29.0368, 200,5.92617, 212,0, 269,0, 280,-3.11035, 300,-
20.9344, 320,-58.7811, 340,-121.734, 360,-214.877;
RMmt 11, 19,24, 3.884, 12, 230,260, 260,290, 0,0, 160,0, 180,-3.43772, 200,-
17.3035, 220,-35.824, 240,-57.8722, 260,-128.545, 280,59.0441, 300,28.4165,
320,13.2492, 340,2.64714, 360,0;
```

Coding 6-2: Revolute joint definitions in MSS input data file

Table 6-5: Corresponding node numbers of respective joints

| Segment (line number) | Theta (rad) | Length (m) |
|---|---|---|
| Foot base (0) | $\pi$ | 0.2508 |
| Foot height (1) | $0.416667\pi$ | 0.06435 |
| Shank (2) | $\pi$ | 0.4059 |
| Thigh (3) | $\pi$ | 0.40425 |
| Abdomen & Pelvic (4) | $\pi$ | 0.198 |
| Thorax (5) | $\pi$ | 0.2772 |
| Neck (6) | $\pi$ | 0.0858 |
| Head (7) | $\pi$ | 0.2145 |
| Upper arm (8) | $\pi$ | 0.3069 |
| Forearm (9) | $\pi$ | 0.2409 |
| Hand (10) | $\pi$ | 0.1782 |

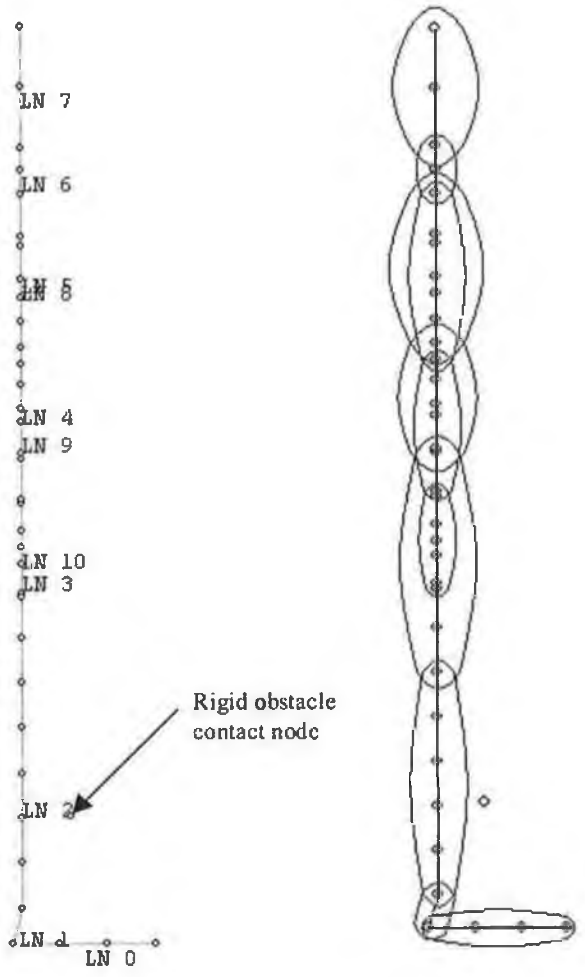


Figure 6-6: Standing human model with rigid obstacle

## 6.3 HUMAN MOTION WITH SIMPLE RIGID OBSTACLE IMPACT

The human body is subject to frontal collision with rigid obstacle at different position of the body and the motion is then studied. The effects from different parameters are analysed with $2^k$ factorial design method.

## 6.3.1 STANDING HUMAN MODEL

The human model described in section 6.2 is used which is of 1.65 m height and of 60 kg mass. The initial position of the human model is defined by the line constraints in Table 6-5. The line constraints will yield the human model to stand straight, which is shown in Figure 6-6. A rigid bar-like obstacle is placed at the position, 0.15 m from the tip of the foot and 0.25 m from the base of the foot. The centre of the obstacle is 0.084145 m perpendicular to the shank. The obstacle has an effective thickness of 0.06 m towards the contact with the shank.

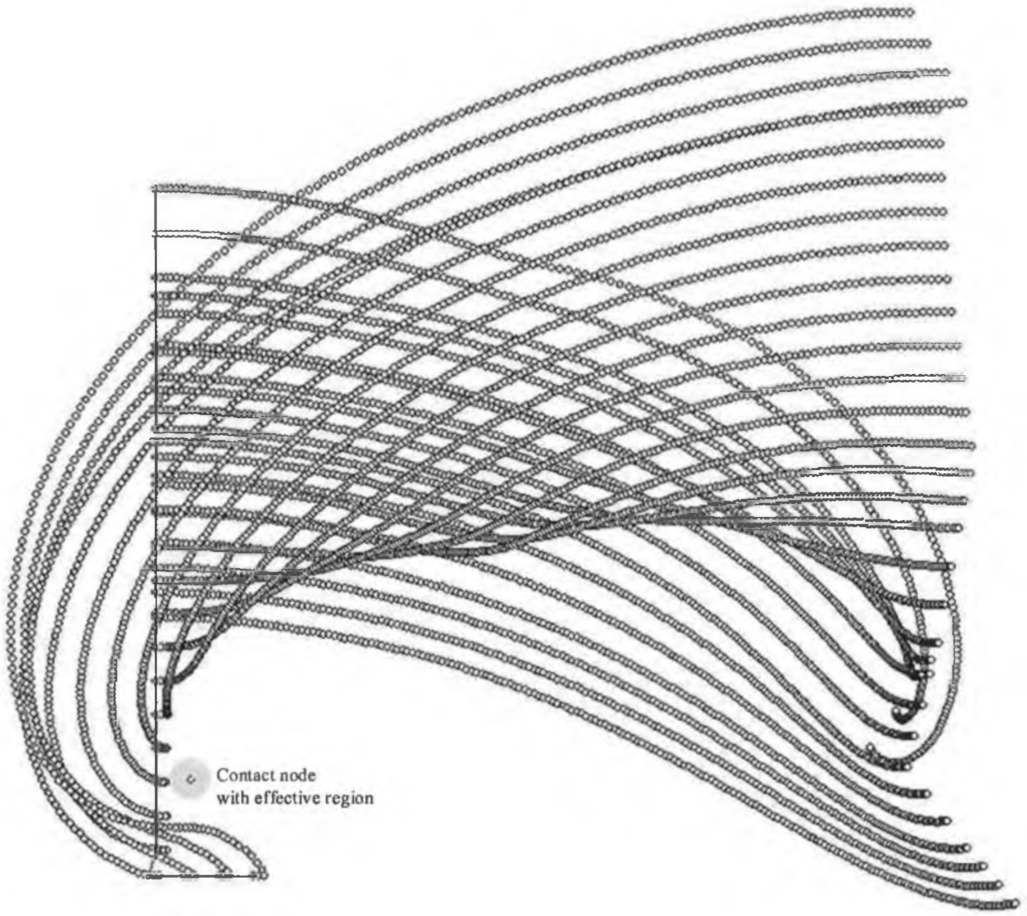


Figure 6-7: Transient response profile of the all nodes in the standing human model problem

The human body is moving with a constant velocity of 10 m/s and the problem is simulated upto 0.25 second. The time step size used is 5 microseconds, while the structural critical time step is 13.4 microseconds. Thus, the problem is simulated for 50,000 time steps, and the processed data are collected at each 250 time steps.

The simulated transient response of the human motion is shown in Figures 6-7 and 6-8. The former is the transient profile of all the nodes. While the latter shows the transient response at every 50 microseconds.

As expected, the human body will topple. The simulation does not incorporate the gravitational effect as it makes negligible influence. The acceleration of the head is monitored. Node 23 is the centre of the head, and its effective acceleration by Least Squared Method (LSM) is recorded. The Head Injury Criterion (HIC) of 10 milliseconds time gap is calculated based on these recorded data. Figure 6-9 shows acceleration histories of node 23. Gx is the LSM acceleration in direction x, Mag(Gx-Gy) is the magnitude of the acceleration, which is calculated with the following equation:

$$\text{Mag}(Gx - Gy) = \sqrt{Gx^2 + Gy^2} \tag{6-1}$$

Two sets of the HIC is calculated based on the Gx alone and magnitude of Gx-Gy.

The maximum HIC index found is very low, 25. The threshold value is 1500 for non-contacting impact on the head where the head is subjected to rapid acceleration/deceleration without any physical collision. Higher value of the HIC index indicates higher risk of head/brain injury. The result shows that the impact at leg (shank) will not create any severe internal injury in the head provided the head is not hit any other obstacle at the later stage. The maximum HIC index of overall effective acceleration is found at about T=0.16625 s. The HIC index of horizontal (x-axis) effective acceleration is found to be maximum at about T=0.15125 s. The HIC of overall acceleration is of course more than the x-axis component. The corresponding positions of the human body are shown in Figures 6-10 and 6-11.

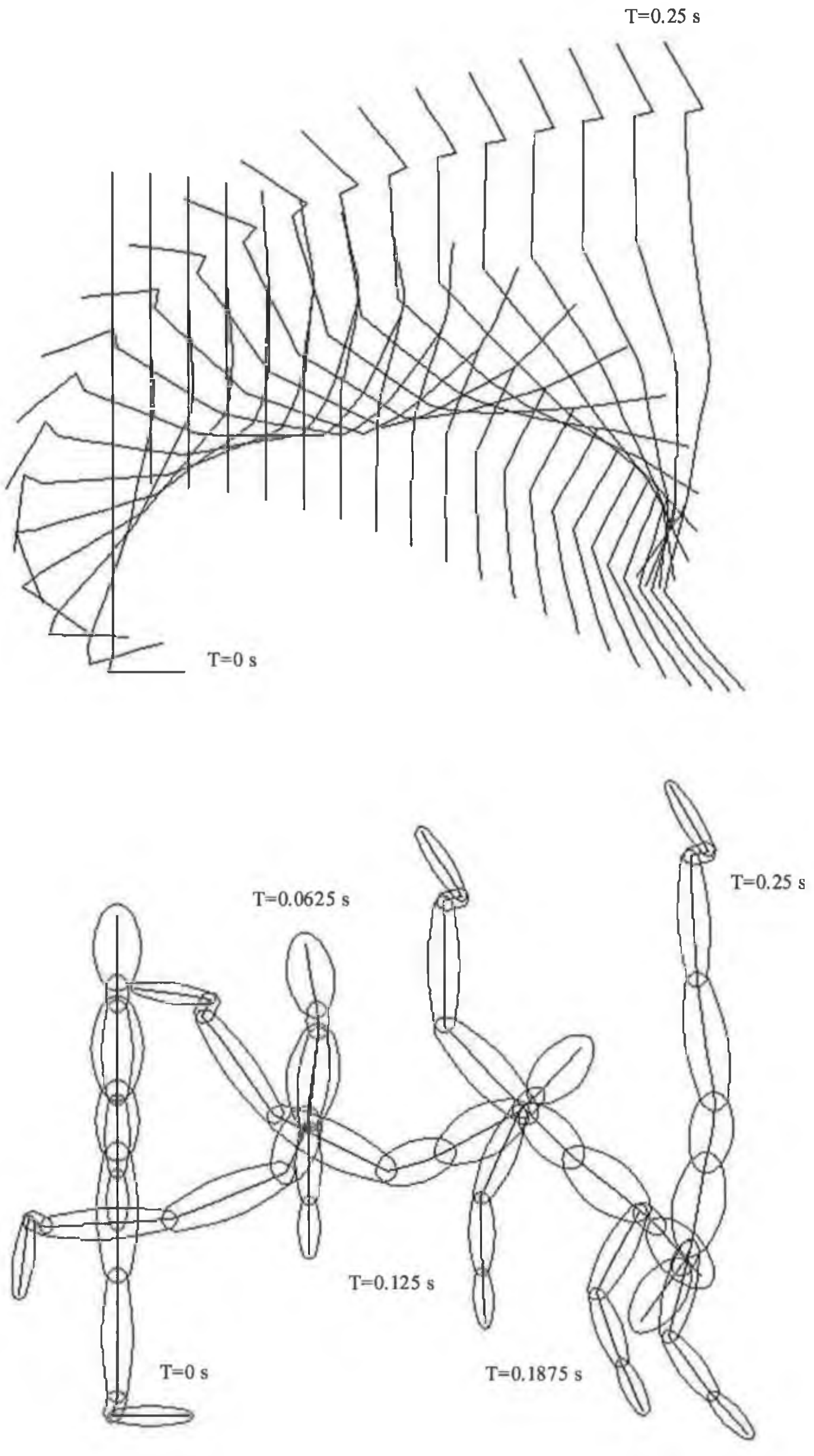


Figure 6-8: Transient response of the standing human model problem

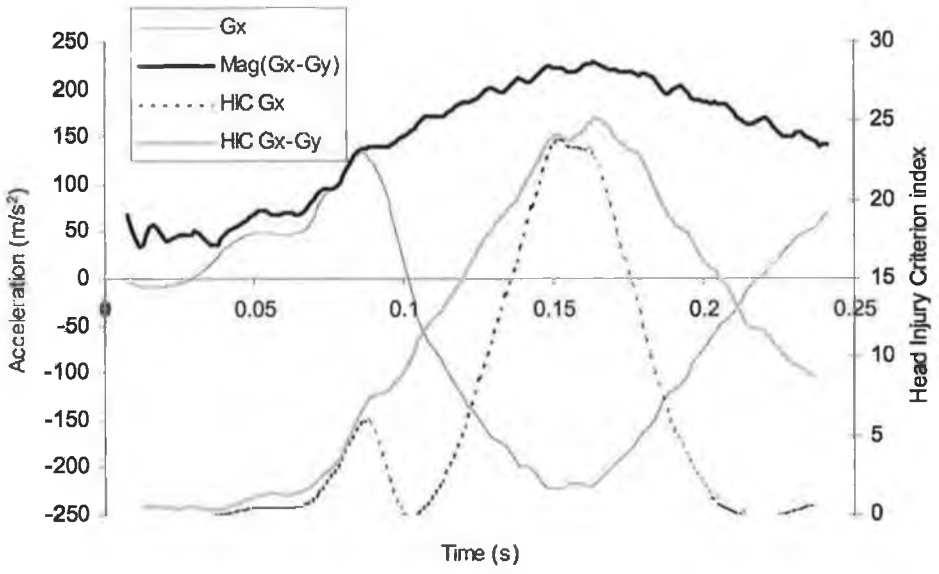


Figure 6-9: Acceleration and HIC history of node 23 in the arbitrary human model problem

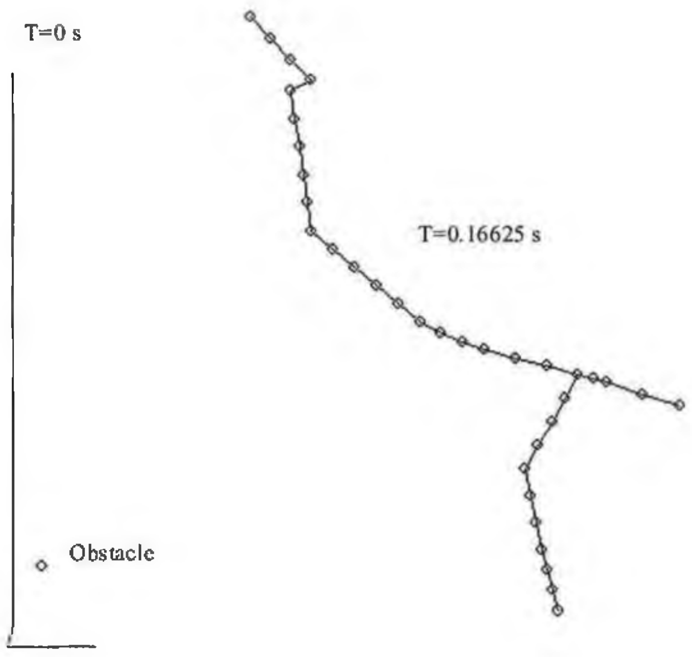


Figure 6-10: Human body position at T=0.11625 second

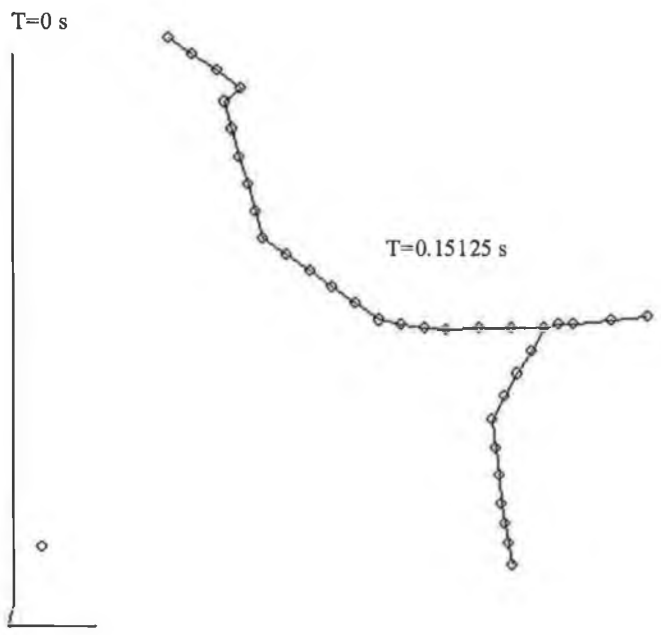


Figure 6-11: Human body position at T=0.15125 second

## 6.3.2 THE $2^4$ FACTORIAL DESIGN OF EXPERIMENT

The human body motion subjected to collision with single rigid obstacle has four main parameters. They are the human height, human weight, impact velocity (human initial velocity), and the obstacle location. The $2^k$ factorial design of experiment is applied to find the parameter(s), which has most significant effect towards the HIC index.

The parameters are divided into two levels, namely high and low. Different parameter has different definition of high and low. For the interest of the study in Asian human body motion, the height of short (low) is set as 1.4 m and tall (high) as 1.8 m. The weight of a human body is defined using Body-Mass Index (BMI). The BMI is defined with the following equations:

$$BMI = \frac{Weight}{Height^2} \tag{6-2}$$

A normal healthy person will have the BMI between 19 and 22. In this study the value of BMI 14 for a very slim person (low in weight) and 30 for an overweight person (high in weight) were used. The corresponding weight of the human model is calculated with this relation after fixing the height.

The main aim of this chapter is to study the motion of the human body subjected to frontal collision in a motorcycle scenario. As mentioned in Chapter 2, an average speed of a motorcycle collision is around 30 km/h. Thus, the low collision velocity was set as 5 m/s, and high collision speed as 20 m/s. The obstacle height is 0.15 m from the ground and 0.70 m from the ground for low and high locations respectively. From the literature (mentioned in section 2.12 of Chapter 2), the legs suffered the highest percentage of motorcycle injuries. Thus the obstacle is placed within the range of the leg, from thigh to shank.

A total of 16 sets of simulation results are required to complete the study. They are consisting of various combinations of the 4 parameters. The abbreviations of various parameters are given in Table 6-6. The following table shows the parametric values of respective experiments. The experiment codes are according to the convention of Montgomery [228].

The simulations use 5 microseconds as the time step size and processed upto either 50,000 time steps with data recorded at every 250 time steps interval or 80,000 time steps with data recorded at every 400 time steps interval. The former arrangement is for the high velocity impact (20 m/s) while the latter is for the low velocity impact (5 m/s). The maximum HIC index of every experiment is recorded, which will be used to measure the overall effect of the impact. 160,000 time steps with data recorded at every 800 time steps interval is used in exception of situation, where it is believed that the maximum HIC occurs at later time, which is more than 0.4 second. They are simulations coded w and h.

The maximum HIC indexes of both Gx and Gx-Gy are recorded in Table 6-8. The graphical profiles of respective simulation are shown in Appendix C.

Table 6-6: Abbreviations used in $2^4$ factorial design

| Abbreviation | Parameter |
|---|---|
| H | Height |
| W | Weight |
| V | Velocity |
| B | Obstacle Height |

Table 6-7: Simulation codes and respective parameter values

| Simulation Code | Height (m) | BMI <Weight (kg)> | Velocity (m/s) | Obstacle height (m) |
|---|---|---|---|---|
| (1) | 1.4 | 14 <29.4> | 5 | 0.15 |
| h | 1.8 | 14 <48.6> | 5 | 0.15 |
| w | 1.4 | 30 <58.8> | 5 | 0.15 |
| v | 1.4 | 14 <29.4> | 20 | 0.15 |
| b | 1.4 | 14 <29.4> | 5 | 0.70 |
| hw | 1.8 | 30 <97.2> | 5 | 0.15 |
| hv | 1.8 | 14 <48.6> | 20 | 0.15 |
| hb | 1.8 | 14 <48.6> | 5 | 0.70 |
| wv | 1.4 | 30 <58.8> | 20 | 0.15 |
| wb | 1.4 | 30 <58.8> | 5 | 0.70 |
| vb | 1.4 | 14 <29.4> | 20 | 0.70 |
| hwv | 1.8 | 30 <97.2> | 20 | 0.15 |
| hvb | 1.8 | 14 <48.6> | 20 | 0.70 |
| wvb | 1.4 | 30 <58.8> | 20 | 0.70 |
| hwb | 1.8 | 30 <97.2> | 5 | 0.70 |
| hwvb | 1.8 | 30 <97.2> | 20 | 0.70 |

Table 6-8: Simulation results

| Simulation | Gx-Gy | | Gx | |
|---|---|---|---|---|
| | Time (s) | Max. HIC | Time | Max HIC |
| (1) | 0.09800 | 0.14410 | 0.31400 | 0.13449 |
| b | 0.06200 | 162.53911 | 0.07000 | 8.88662 |
| vb | 0.02375 | 22657.68488 | 0.02375 | 19015.75470 |
| v | 0.12500 | 71.73064 | 0.11375 | 70.97219 |
| wv | 0.13250 | 97.98397 | 0.11875 | 93.64851 |
| wvb | 0.02625 | 34651.57348 | 0.02625 | 28428.60670 |
| wb | 0.06000 | 33.67115 | 0.07000 | 17.93349 |
| w | 0.34400 | 0.20937 | 0.37200 | 0.20334 |
| h | 0.44800 | 0.09047 | 0.46800 | 0.08518 |
| hv | 0.14875 | 56.77950 | 0.15000 | 56.17742 |
| hvb | 0.04500 | 6648.43947 | 0.04625 | 5823.96161 |
| hb | 0.04800 | 6.31175 | 0.04800 | 5.60660 |
| hwb | 0.08600 | 10.95503 | 0.11400 | 6.33046 |
| hwvb | 0.03875 | 9135.26228 | 0.04125 | 7572.98509 |
| hwv | 0.15875 | 113.47247 | 0.16000 | 107.53143 |
| hw | 0.28600 | 0.06766 | 0.28600 | 0.06766 |

From the table shown, one can easily observe the combination of high-speed (v) with high obstacle level (b) will cause serious impact towards the head. While the rest is within the HIC threshold value. The maximum value of Gx-Gy HIC index is used to find the most "effective" parameter towards head acceleration/deceleration. An effective index is calculated based on the following equation:

$$X_{eff} = \frac{X_{contrast}}{16} \tag{6-3}$$

where the contrast of $X$ parameter is given as:

$$X_{contrast} = \sum HIC_{i,X} \times const_{i,X} \tag{6-4}$$

The $HIC_i$ is maximum HIC index of parameter $i$. While the $const_i$ is contrast constant, which is given in Table 6-9. The subscript $X$ is the indication of parameter $X$, which is to be calculated. The effect indexes of various parameters are calculated and shown in Table 6-10.

Table 6-9: Contrast constants for $2^4$ factorial design

| | H | W | HW | V | HV | WV | HWV | B | HB | WB | HWB | VB | HVB | WVB | HWVB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (1) | -1 | -1 | 1 | -1 | 1 | 1 | -1 | -1 | 1 | 1 | -1 | 1 | -1 | -1 | 1 |
| h | 1 | -1 | -1 | -1 | -1 | 1 | 1 | -1 | -1 | 1 | 1 | 1 | 1 | -1 | -1 |
| w | -1 | 1 | -1 | -1 | 1 | -1 | 1 | -1 | 1 | -1 | 1 | 1 | -1 | 1 | -1 |
| hw | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 | 1 |
| v | -1 | -1 | 1 | 1 | -1 | -1 | 1 | -1 | 1 | 1 | -1 | -1 | 1 | 1 | -1 |
| hv | 1 | -1 | -1 | 1 | 1 | -1 | -1 | -1 | -1 | 1 | 1 | -1 | -1 | 1 | 1 |
| wv | -1 | 1 | -1 | 1 | -1 | 1 | -1 | -1 | 1 | -1 | 1 | -1 | 1 | -1 | 1 |
| hwv | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| b | -1 | -1 | 1 | -1 | 1 | 1 | -1 | 1 | -1 | -1 | 1 | -1 | 1 | 1 | -1 |
| hb | 1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 | 1 | -1 | -1 | -1 | -1 | 1 | 1 |
| wb | -1 | 1 | -1 | -1 | 1 | -1 | 1 | 1 | -1 | 1 | -1 | -1 | 1 | -1 | 1 |
| hwb | 1 | 1 | 1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 | 1 | -1 | -1 | -1 | -1 |
| vb | -1 | -1 | 1 | 1 | -1 | -1 | 1 | 1 | -1 | -1 | 1 | 1 | -1 | -1 | 1 |
| hvb | 1 | -1 | -1 | 1 | 1 | -1 | -1 | 1 | 1 | -1 | -1 | 1 | 1 | -1 | -1 |
| wvb | -1 | 1 | -1 | 1 | -1 | 1 | -1 | 1 | -1 | 1 | -1 | 1 | -1 | 1 | -1 |
| hwvb | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Table 6-10: Parameter(s) effect indexes

| Parameter | Effect Index |
|---|---|
| HW | -583.95019 |
| HWB | -587.74413 |
| HWV | -600.62808 |
| HWVB | -604.44405 |
| WB | 892.093624 |
| W | 902.467217 |
| WVB | 907.627015 |
| WV | 917.989995 |
| HV | -2584.1175 |
| HVB | -2584.2091 |
| H | -2606.5099 |
| HB | -2606.5526 |
| VB | 4533.75176 |
| B | 4560.37244 |
| V | 4576.18363 |

The symbol "H" in Table 6-10 refers to the effect of parameter height, HW refers to the effect of height and weight interaction and so on. The result shows that V, B and VB interaction play a very important role in influencing the impact severity towards the human head. A rider has to sustain higher risk of head/brain injury when subjected to higher velocity of collision and higher height of contact location. Following the height of a person and the weight of a person contributes the least.

Combined effects of height and weight are not influencing significantly relative to the velocity of collision and the obstacle location. One can conclude that the height and weight of a person are not as important as the impact velocity while subjected to any kind of obstacle impact. It is worth noting in this finding that the combined contribution of all the parameters is among the lowest.

## 6.4 HUMAN WITH MOTORCYCLE CRASH SIMULATION

It is evident from the literature (section 2.11.3) shows that 40.1% of the motorcycle crashes are frontal impacts. An approximated model of Malaysia national motorcycle, KRISS 110, by Modernas Sdn. Bhd., is developed. The model is simplified and approximated. Figure 6-12 shows a schematic diagram of the motorcycle. Table 6-11 shows the overall dimensions and the weights of the KRISS 110. Descriptions of the KRISS motorcycle can be found in Figure D-1 and Figure D-2 of Appendix D.

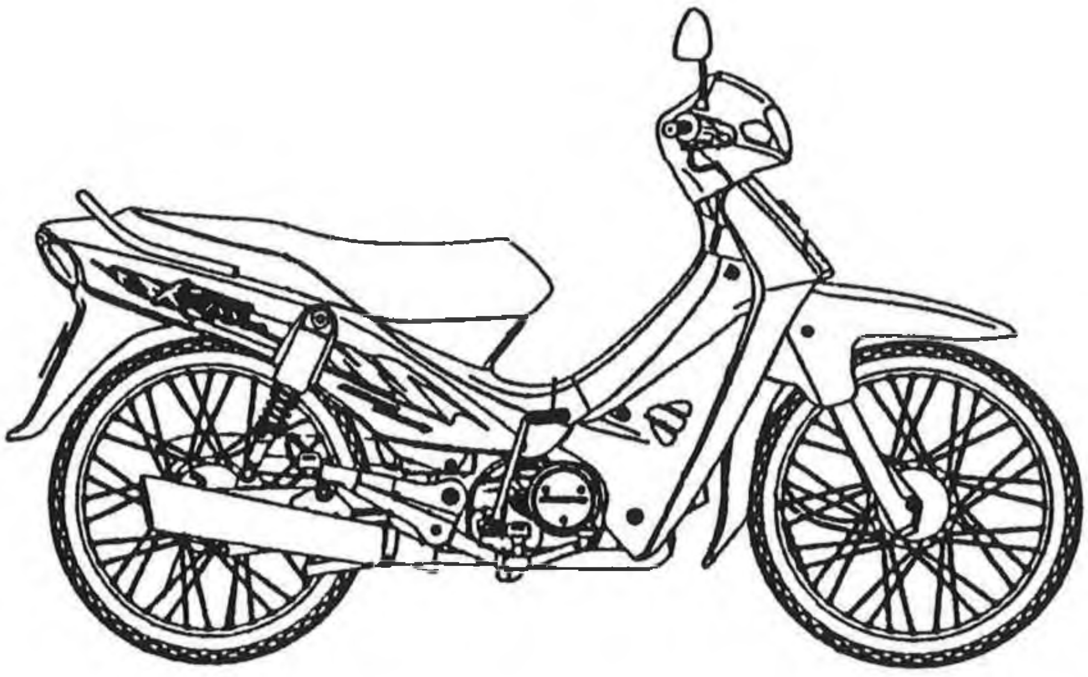


Figure 6-12: Modernas KRISS 110

Table 6-11: Modernas KRISS 110 overall dimensions and weights

| | |
|---|---|
| Overall length | 1910 mm |
| Overall width | 670 mm |
| Overall height | 1050 mm |
| Wheel base | 1245 mm |
| Road clearance | 135 mm |
| Weight with empty tank | 98 kg |
| Weight with full tank | 100 kg |

The motorcycle is modelled with several deformable bars, except the wheels, which are not modelled physically. Figure 6-13 shows the discretised model. The corresponding dimensions of the links are shown in the following Table 6-12. Figure 6-14 shows the stress-strain relation of the material used for the motorcycle structure. The material is idealised carbon steel with the density of 7850 kg/m$^3$. The actual strength in bending for each component is unknown and the thickness of the link was selected arbitrarily to give certain bending strength, which would be realistic.

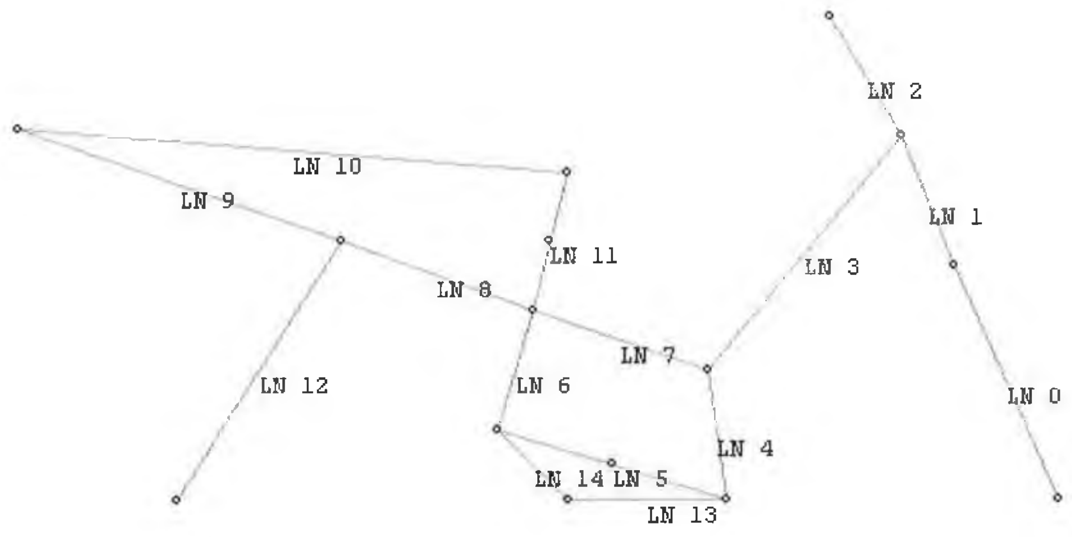


Figure 6-13: Discretised KRISS 110 model

Table 6-12: Parameters of the KRISS 110 model

| Ln | Length | Width | Height |
|---|---|---|---|
| 0 | 0.36441 | 0.01748 | 0.1 |
| 1 | 0.19925 | 0.01598 | 0.2 |
| 2 | 0.19847 | 0.03209 | 0.1 |
| 3 | 0.53393 | 0.01484 | 0.2 |
| 4 | 0.18668 | 0.03412 | 0.1 |
| 5 | 0.51798 | 0.10113 | 0.15 |
| 6 | 0.77947 | 0.03555 | 0.1 |
| 7 | 0.80340 | 0.02438 | 0.1 |
| 8 | 0.17915 | 0.02207 | 0.1 |
| 9 | 0.28857 | 0.00527 | 0.1 |
| 10 | 0.48936 | 0.00327 | 0.1 |
| 11 | 0.55470 | 0.01255 | 0.1 |
| 12 | 0.25986 | 0.01455 | 0.1 |
| 13 | 1.55000 | 0.01146 | 0.1 |
| 14 | 0.65000 | 0.01823 | 0.1 |

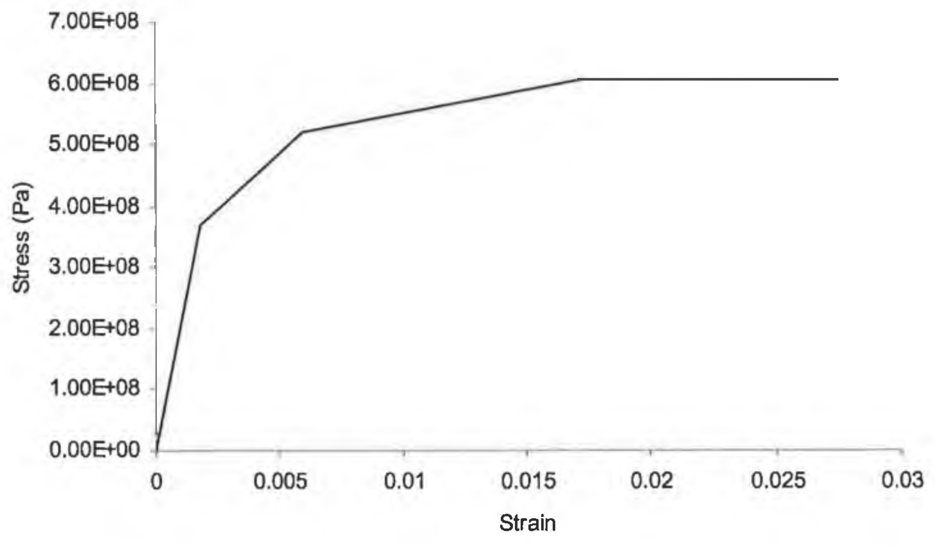


Figure 6-14: Stress-strain relation for motorcycle structure

A human model as described in Section 6.2 is incorporated into the model. The human model is placed as in a normal sitting posture. Rigid barriers are placed both horizontally and vertically. The former barrier is the ground and the latter is the impact barrier. The initial set-up of the model is shown as Figure 6-15.

Contact-impacts occur, between the motorcycle and the ground, between the motorcycle and the front barrier, and between the human model with the motorcycle. For the time being, the contacts between human and motorcycle, are limited to three areas, pelvic-seat, foot-foot rest, and human body-handle bar. Frictionless KCA/AP Contact-impact elements are used to handle the events. Coding 6-3 shows the contact-impact element definitions and Figure 6-16 shows the corresponding element members (the contact node and the contact link/surface).

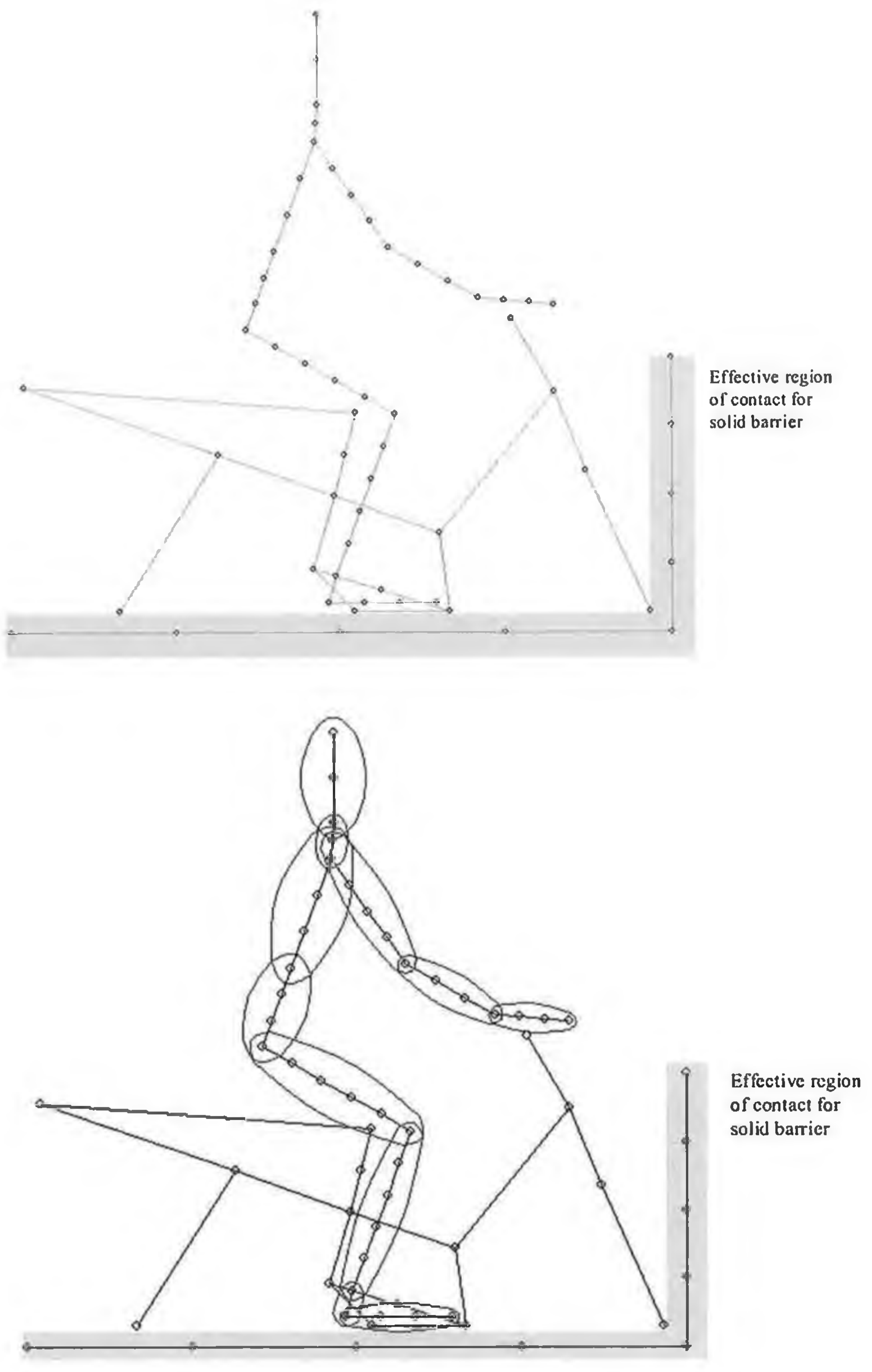


Figure 6-15: Human-motorcycle model

```
Ctact 0, 35, 53, 6, 0, 5e+007, 0, 0.15, 0.9, 0.05, 0;
Ctact 1, 35, 54, 6, 0, 5e+007, 0, 0.15, 0.9, 0.05, 0;
Ctact 2, 35, 55, 6, 0, 5e+007, 0, 0.15, 0.9, 0.05, 0;
Ctact 3, 35, 56, 6, 0, 5e+007, 0, 0.15, 0.9, 0.05, 0;
Ctact 4, 35, 57, 6, 0, 5e+007, 0, 0.15, 0.9, 0.05, 0;
Ctact 5, 37, 31, 6, 0, 5e+007, 0, 0.15, 0.9, 0.02, 0;
Ctact 6, 37, 32, 6, 0, 5e+007, 0, 0.15, 0.9, 0.02, 0;
Ctact 7, 37, 11, 6, 0, 5e+007, 0, 0.15, 0.9, 0.05, 0;
Ctact 8, 37, 12, 6, 0, 5e+007, 0, 0.15, 0.9, 0.05, 0;
Ctact 9, 37, 13, 6, 0, 5e+007, 0, 0.15, 0.9, 0.05, 0;
Ctact 10, 37, 14, 6, 0, 5e+007, 0, 0.15, 0.9, 0.05, 0;
Ctact 11, 37, 15, 6, 0, 5e+007, 0, 0.15, 0.9, 0.05, 0;
Ctact 12, 37, 16, 6, 0, 5e+007, 0, 0.15, 0.9, 0.05, 0;
Ctact 13, 14, 45, 6, 0, 5e+007, 0, 0.15, 0.9, 0.1, 0;
Ctact 14, 48, 0, 6, 0, 5e+007, 0, 0.15, 0.9, 0.02, 0;
Ctact 15, 48, 1, 6, 0, 5e+007, 0, 0.15, 0.9, 0.02, 0;
Ctact 16, 48, 2, 6, 0, 5e+007, 0, 0.15, 0.9, 0.02, 0;
Ctact 17, 47, 51, 6, 0, 5e+007, 0, 0.15, 0.9, 0.05, 0;
Ctact 18, 47, 52, 6, 0, 5e+007, 0, 0.15, 0.9, 0.05, 0;
Ctact 19, 36, 57, 6, 0, 5e+007, 0, 0.15, 0.9, 0.05, 0;
Ctact 20, 36, 58, 6, 0, 5e+007, 0, 0.15, 0.9, 0.05, 0;
Ctact 21, 38, 57, 6, 0, 5e+007, 0, 0.15, 0.9, 0.05, 0;
Ctact 22, 38, 58, 6, 0, 5e+007, 0, 0.15, 0.9, 0.05, 0;
```

Coding 6-3: Contact element definitions

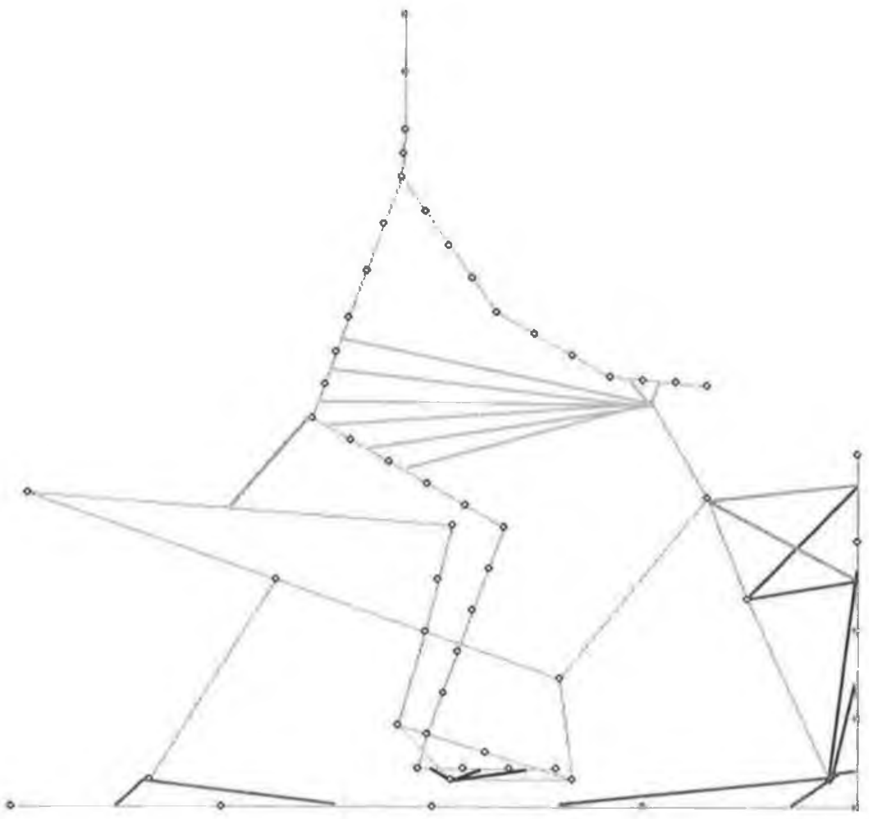


Figure 6-16: Contact elements of the human-motorcycle model

The whole human-motorcycle moves with 10 m/s as the initial velocity. The model is processed with the MSS Node-Layer element. The time step size is set to 5 microseconds, the simulation was continued for 80,000 time steps, which is equivalent to 0.4 second. The gravitational effect is ignored.

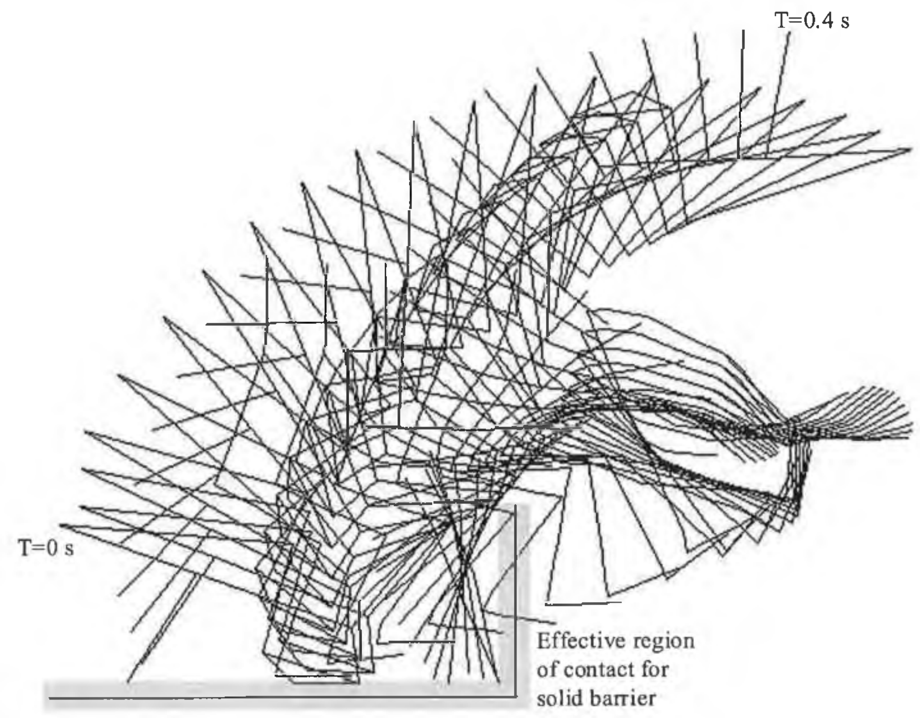
Figure 6-17 shows the overall solution of the simulation. The time interval between two frames is 0.02 second. Obviously, the motorcycle is toppled over the frontal rigid barrier. The human is well trapped at the handle bar.

This is somewhat not realistic, as the human normally experience the first impact at the legs. This is more clearly shown in Figure 6-18. The human will hit the motorcycle's structure at time from 0.04 second to 0.05 second. This is due to the lack of detail motorcycle structure modelling, as well as detail contact-impact definitions. The motorcycle structure such as the front carrying basket and cowl can be incorporated into the motorcycle modelling.

The motorcycle is toppled during the impact, which can be seen even at the initial stage of the impact. Crash test by TRL[*] shows similar characteristic. Figure E-1 of Appendix E shows a clip of the motorcycle crash test by TRL. The bending strength of the front fork of the model developed in this study has extra strength compared to the experimental results. Anyhow, the comparison is not valid, as the motorcycle used at TRL was not modelled due to lack of information of the experimental parameters.

A more detail motorcycle model is developed and is shown in Figure 6-19. The obstacle due to the frontal fairing of the motorcycle is handled with the new model and additional contact elements are defined. The new configuration was simulated upto 0.525 second with consideration of the gravitational effect ($g=9.81m/s^2$). Figure 6-20 shows the transient response of the human-motorcycle model. As the first contact, the knees hit the obstacle due to the fairing of the motorcycle. The shanks are then in contact with the fairing and sliding occurs.

[*] British Transport Research Laboratory, Crowthorne, RG45 6AU, UK.



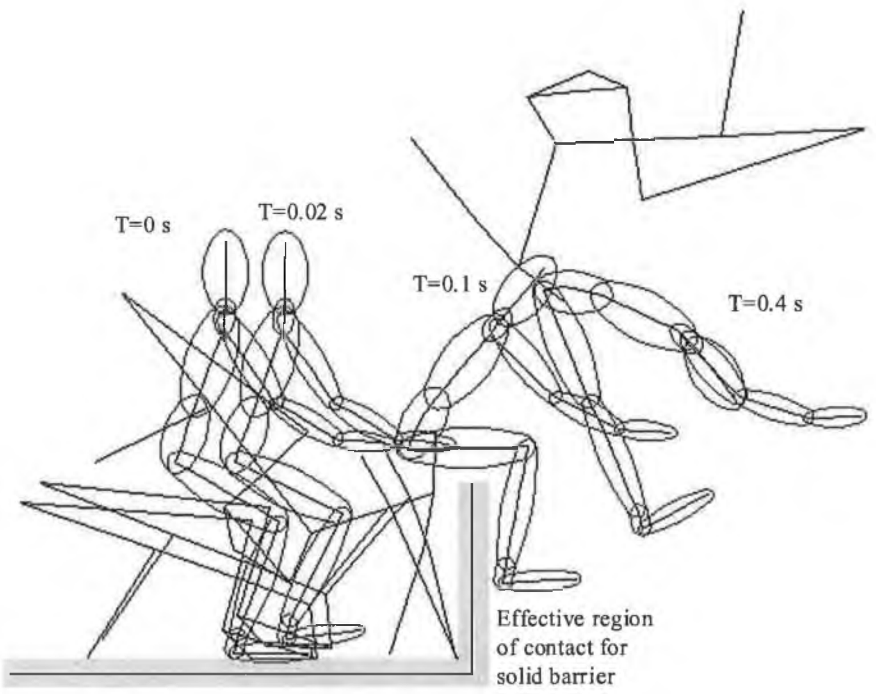


Figure 6-17: Transient response of the human-motorcycle model (above: time interval of 0.02 second between frames)

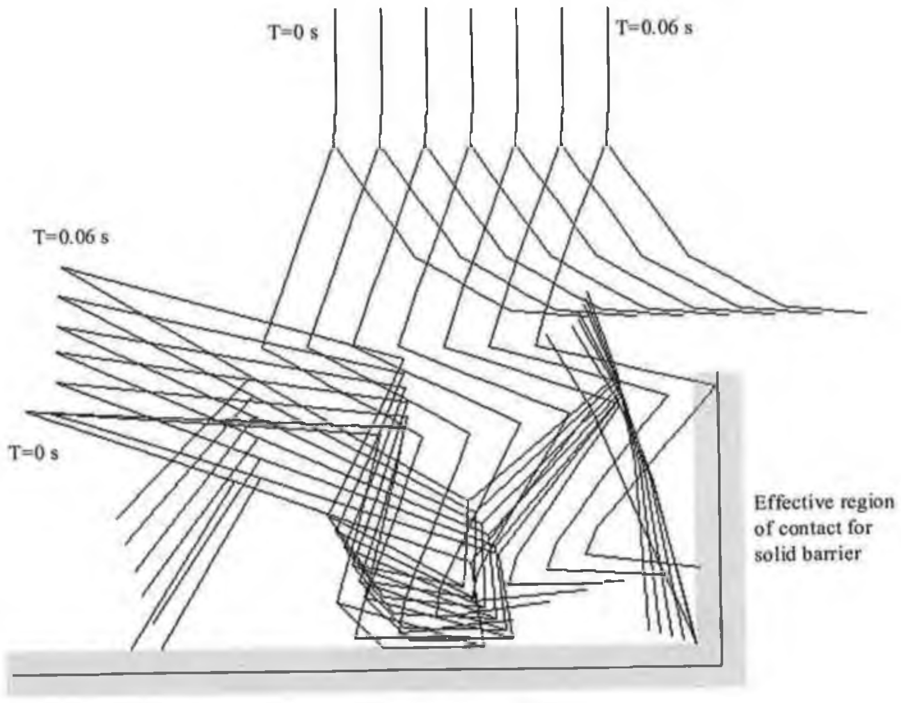


Figure 6-18: Impact at initial stage

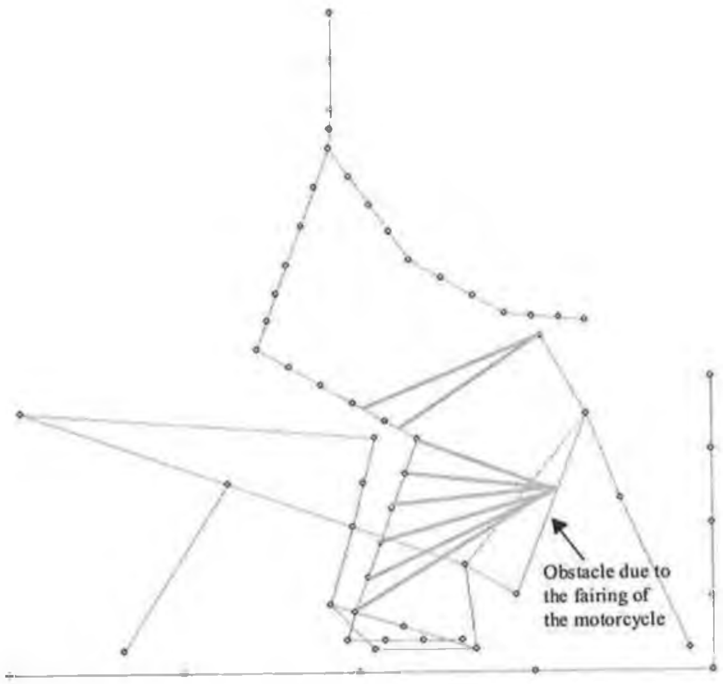


Figure 6-19: Human-motorcycle model with obstacle due to the fairing and definition of extra contact elements.

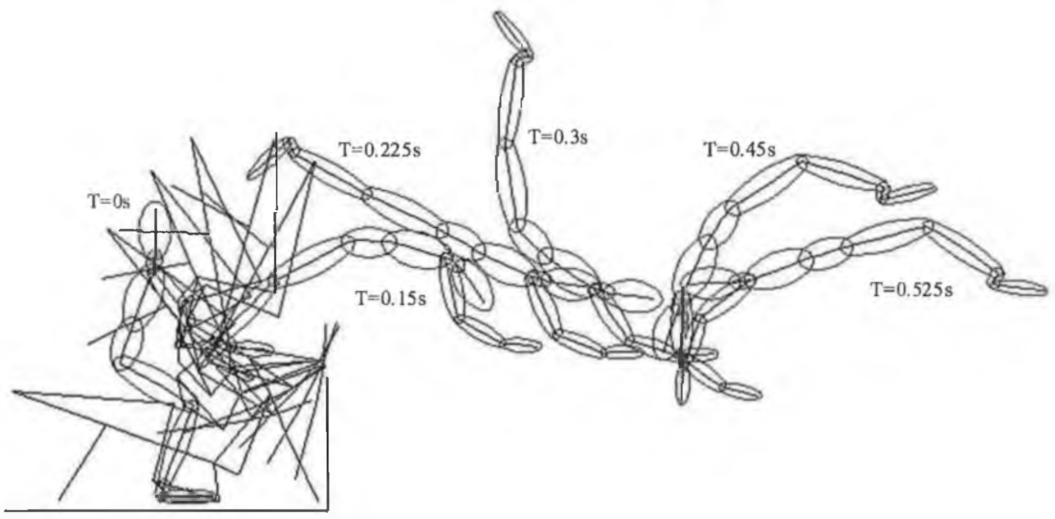


Figure 6-20: Transient response of the new human-motorcycle model

Owing to the toppling of the motorcycle as well as the sliding of the shanks, the human model is being pushed upwards. The next contact occurs between the thighs and the handle bar, which provides additional rotary motion to the human model. Thus, the human model is "thrown" horizontally with rotary motion before reaching the ground.

The prediction of the new model is more realistic when compared to the initial human-motorcycle model. This also shows that the first contact location between the human and the motorcycle influences the overall motion of the human body significantly.

## 6.5 CONCEPTUAL DESIGN OF SLED TEST RIG

It is not always appropriate to crash complete vehicles in expensive full-scale tests. This is particularly true when investigating specific vehicle components under varying impact conditions. It is also more difficult to control impact conditions when testing smaller vehicles and investigating the effects of impacts on people.

The sled test rig can be used to investigate the effects of impacts on a variety of objects besides motorcycle, such as bicycles, scooters and trailers. Besides that, seats, seat belts, seat mountings, air bags, steering wheels, child restraints, car body shells, absorbers, bumper energy, leg protectors, car windscreens, etc, can be investigated in different crash modes. Human dummies can be used during the tests.

The sled test rig offers the advantages of simplicity, control of impact conditions, accurate data collection, excellent repeatability, and a safe working environment. Furthermore, each run of the sled test is far less expensive than corresponding full-scale test. As mentioned in Chapter 2, the average cost of a full-scale crash test is around 20,000 pounds Sterling and 2,000 pounds Sterling for a sled test. These costs exclude the initial investment for the rig.

The sled test rig in TRL comprises a sled running on rails which is towed to the impact speed by a falling weight acting through a system of pulleys. Objects under test can be mounted on the sled and subjected to controlled deceleration by an arrestor. Alternatively, they can be mounted on a rigidly fixed frame and impacted by projectiles released by the sled. The TRL's rig is capable of producing controlled impact speeds of upto 70 km/h and controlled deceleration of up to 70g.

## 6.5.1 FUNCTIONS AND REQUIREMENTS

The main functions of the sled test rig are:

- Perform human dummy motorcycle crash tests
- Perform other structural and vehicle component impact tests

To achieve such functions, several preliminary requirements of the test rig are:

- Generate controlled-speeds of up to 20 m/s or more
- Support loads up to 350 kg
- Support the controlled-deceleration, which minimises all kinds of post-impact failures of any components
- High repeatability and reliability
- Fast speed image capturing facility

### 6.5.2 MAIN COMPONENTS

Figure 6-21 shows an overview of a crash test rig. The rig consists of guide rails, shove cylinders, moving platform, fixtures, and collision block with deceleration element.

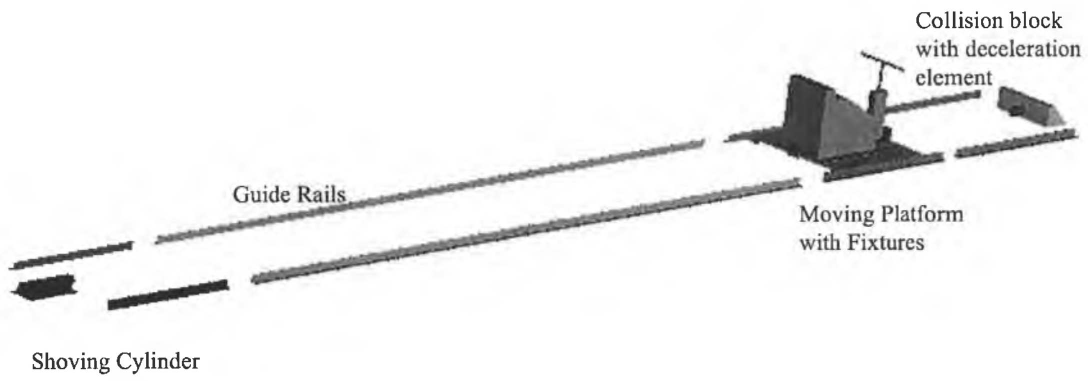

Collision block with deceleration element
Guide Rails
Moving Platform with Fixtures
Shoving Cylinder

Figure 6-21: Overview of a sled test rig

The shoving cylinder accelerates the moving platform to the required speed and have to be a powerful pneumatic cylinder. The platform is guided by rails on to the collision block. The collision block is equipped with deceleration element, which is used to stop the moving platform. The deceleration element may consist of springs, stacked rings, etc, which will produce different crash modes.

Figure 6-22 gives a closer look of the moving platform of the sled test rig. Air bearings are used to move the platform. Air bearing gives virtually frictionless movement, which can reduce the burden of shoving cylinder in achieving higher speed. For the guide rails, standard L-shape structural steels can be used. The fixtures showed are not of its final design. The fixtures must be able to accommodate the real motorcycle crash situation, thus, it should be possible to position a real motorcycle on the platform. Certain characteristics have to be incorporated as well, such as toppling effects during the impact. Fixtures are used to hold the necessary components, such as motorcycle and human dummy. Figure E-2 in Appendix E shows the fixtures and amended motorcycle with human dummy used at TRL.

Fast speed cameras are required to capture the motion during the collision. A 1000 frames per second speed will be sufficient to capture the motion, provided the shutter speed must be fast enough to capture "still" image. This can be observed from the simulated results, where 0.02 second interval motion frame is sufficient to be used for study (refer to Figure 6-17).

To run the test, a good human dummy is very important. Anyhow, standardised dummies are available in the market. A Hybrid III dummy will be a good choice. Customisations may be required, especially in both the hands, which will grab the handle bar of the motorcycle. Furthermore, extra sensors may be required on both the legs, as most of the frontal collisions involve legs. Again, the cost of a commercially available human dummy is very expensive. Detail engineering drawing of the rig can be found in the Appendix F.

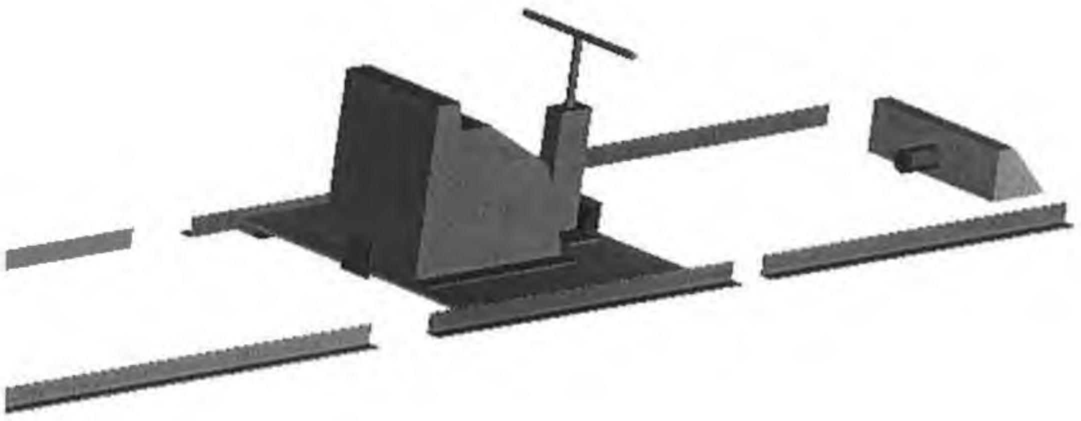


Figure 6-22: Close view at the moving platform of a sled test rig

## 6.6 DISCUSSIONS AND SUMMARIES

The capability of the MSS in modelling human model subjected to crash impact has been demonstrated. The human model development is explained at the beginning of the chapter. A better representation can be carried out with the understanding of human anatomy as well as human responses when subjected to "excitement". Two-dimensional elements may be used instead of the one dimension Node-Layer element, to model the human body. This will give more realistic representation in term of the dimension. But, it will take up much more computational effort to solve similar problem. Thus, one-dimensional element could be more appropriate to be used in simulating simplified crash scenario involving human.

It has been a fact that human will push himself/herself towards the highest limits of respective physical limits when in critical situation. During impact, the human muscular responses play an important role. Knowledge in this field is still scarce. The conventional resistive moment of the joints is not enough to represent the response of the muscles, especially in crash situation. Work in this area of biomechanics is needed to give a clearer picture towards a better human modelling representation. An arbitrary human model subjected to collision with a rigid obstacle at the shank is shown. The transient motion profile is displayed and the HIC index profiles are shown as well.

Preliminary studies are carried out in human frontal collision, dealing with different impact parameters, which are the human height, human weight, collision speed, and obstacle location. Simulations of various combinations of the parameters were conducted, according to the $2^4$ factorial design of experiment. The purpose is to find the most significant effect or combined effects towards the head acceleration/deceleration during the collision. The study found that the combined effect of human height and human weight give the least contribution in head injury severity. The location and the velocity of the collision appear to be the most significant independent effects as well as combined effect among the parameters.

An idealised motorcycle model is developed based on the KRISS 110, a Malaysian national motorcycle. The model consists of several one-dimensional elements.

Assumptions are made during the development of the model, due to the lack of information. They are the mass distribution and the bending strength between links. This information can only be gathered with the collaboration of the Modernas, the manufacturer of the KRISS, and by conducting experiments. The human model is incorporated into the model successfully and the solution was found to be satisfactory. The model can produce more sound results with more accurate representation, especially in the contact-impact element definitions and motorcycle physical dimension modelling.

# Chapter 7: Discussion & Conclusions

## 7.1 Introduction

The work of this study is summarised and concluded in this chapter. The contributions arising from the thesis are presented following the conclusions, along with the recommendations for further work.

## 7.2 Discussion

A number of conclusions may be drawn from the work done in this study in several areas, from the development of the two generations of simulation systems to the incorporation of the novel support elements. Preliminary numerical studies have been carried out on human motion subjected to frontal collision, both with rigid obstacles and when riding a motorcycle.

### 7.2.1 DEVELOPMENT OF THE FDM

The FDM is the first simulation system developed in this work. The system incorporated pre-, post-, and the main-processor in a single system. A pre-processor input data file format is suggested to deal with the FD method-based numerical model. The file format is capable of storing information about simple geometry structure in line format. The meshing parameters are provided along with the line definition. The boundary conditions, contact-impact information, and revolute joint definitions can be specified. The FDM pre-processor is, by and large, a primitive system. All information can only be fed through the pre-processor input file via a text editor. Meshing (generating nodes and links) is automated in FDM pre-processor accordingly. 1-D meshing algorithm is successfully implemented into the FDM pre-processor.

The FD model of Witmer *et al.* [1] and Hashmi *et al.* [28] has been successfully implemented in the FDM processor. A generalised input data file of the FDM processor, which is generated by the FDM pre-processor has been proposed. It is proved to be capable of handling most of the modelling situations within the limitation of the numerical model. "Elementary" contact-impact is considered, where contact-impact of the structure onto a rigid node (can be treated as a solid bar) has developed and implemented successfully. Simple revolute joint element, which is only suitable for the adopted model, is suggested and successfully implemented.

The FDM post-processor is capable of presenting the solution from the FDM processor. Animated transient response displaying changing contour of the structure has been developed.

The development of the FDM was based on a dedicated hydrocode used to study the response of stacked rings subjected to impact. The FDM is limited for future expansion both in terms of numerical applications and the development to cope with more different types of elements. The limitation of the FDM is mainly due to the incorporated FD model, as well as, the construction and development of the system itself.

The initial development of the numerical method based simulation system, has led to the development of new element formulations, new hydrocodes and new system.

## 7.2.2 DEVELOPMENT OF THE MSS

The MSS is the second generation of numerical technique based simulation system developed in this study. As mentioned above, the restructure of the FDM is necessary for further expansion. Thus, a new system has been developed in which object-oriented design is integrated in dealing with both the element definitions and implementations. All the MSS components are developed from fresh without any reference to the FDM, and has proven to be a better environment for development and implementation. The method makes the design and development of all components (pre-, post- and the processor) easier and systematic. All the three components are independent, which is different from the FDM. They were developed as separate applications. This enabled the user to work on the same or different model in the MSS pre-processor or viewing processed data with the MSS post-processor while the MSS processor is processing the model.

The MSS pre-processor is a better environment to work on compared to the FDM pre-processor. A simple 2-D CAD environment is developed to work on the pre-processing tasks. Furthermore, all modelling facilities can be found in the MSS pre-processor. As the result, the user can develop a model from fresh in the MSS pre-processor without entering any "code" as required in FDM pre-processor. The geometry entities and modelling entities can be defined easily with graphical user interface. They are encapsulated under the respective classes. The object-oriented programming (OOP) technique is used to handle all the entities. The OOP once again proven to be a better programming technique compared to the conventional structured programming in developing huge code. Data files for different purposes are suggested. The text-based data file is for the convenience of direct interpretation, and for amendment by the user. While, the binary data file is for the convenience of the system to read and write, as well as for storage efficiency. One main drawback of the MSS Pre-processor is the repeated tedious tasks to work on especially editing different conditions. Defining and editing contact elements are also tedious tasks in certain cases. To improve the situation, additional automated algorithms need to be designed and developed.

The OOP technique is incorporated in the MSS processor development as well. Definitions and implementations of all entities used in MSS processor, such as the one-dimensional structure elements, nodes, and contact elements. Multithreads are used in the MSS processor. This makes the utilisation of the computer resources more efficiently. This has proved to be a better way.

With the object-oriented design, the possibility of having a systematic way to deal with numerical technique based formulation especially with the finite difference method was found, which may lead to a better methodology than the finite element method (FEM). Easier understanding, formulation, and avoidance of large matrixes involved are the main advantages of the object-oriented designed method comparing to the FEM.

The MSS post-processor has the similar features as the FDM post-processor. The significant differences are the capability of the MSS post-processor to generate customisable text-based output data files and the lacking of contour plotting in the MSS post-processor.

In general, the MSS is a capable system in dealing with impact studies. All the simulation examples and numerical problems in Chapter 5 and Chapter 6 are solved using the MSS. The MSS can serve as a tool to be used. Further amendments and improvements are required to make the MSS a versatile system, which can incorporated any type of elements, including 3-D element.

### 7.2.3 FDM STRUCTURAL ELEMENT

The FDM model is a well-known general model and has been used successfully in solving many numerical dynamic-impact problems. The numerical method is an Eulerian code. It is simple in both understanding and implementation. However, the method suffers several limitations despite its generality, which have been mentioned in section 5.2.4 of Chapter 5.

### 7.2.4 MSS STRUCTURAL ELEMENTS

Two new elements, which have been incorporated in the MSS, have been developed. Both the elements have the same capability, but are different in implementation method and concept in handling the bending with the neighbour elements. They are called Bending-Division element and Node-Layer element respectively. Both methods eliminated the shortcomings of the model of references [1] and [28]. They are more suitable to be used as general elements. Comparisons between the previous numerical method and the new methods have been made. All three methods were used to solve impact of triangular frames. The results showed that all the three methods correlate each other.

Comparison of the responses from all the three methods in solving a T-structure was made. Special treatments of Witmer *et al.* and Hashmi *et al.* model is described, and is very inconvenient, both in developing the model and implementation of processing tasks. Good correlation was obtained among the three methods in solving the T-structure problem. The Bending-Division method gave more resistance compared to the Witmer *et al.* and Hashmi *et al.* method and Node-Layer method. The latter two methods gave very close responses. The Node-Layer element has been used to simulate impact of a tapered cantilever. Comparison with the experimental result was made and very good correlation was achieved.

The Bending-Division method is more efficient in terms of implementation, which is shown by the comparison of the execution time among the three methods in solving T-structure. But, the Node-Layer method also makes reasonable comparison. With a good prediction and a small trade-off of computational time, the Node-Layer method is recommended.

### 7.2.5 CENTRAL DIFFERENCE TEMPORAL INTEGRATION SCHEME

Central difference temporal integration scheme is proven to be a good explicit temporal integration scheme. It has been successfully incorporated it in both the FDM and the MSS dealing with various numerical components, such as the structural element, contact element, etc.

### 7.2.6 NUMERICAL STABILITY

As no stability proof is available for time integration of the non-linear problem, a safety factor of 0.9 is used for the stability criterion. Avoidance of the occurrence of instability would be improved with better understanding of the instability occurrence. The reason of instability has been explained earlier in the thesis and the 0.9 safety factor is good enough for the time being, as no element instability has been evident.

### 7.2.7 MATERIAL MODELLING

A versatile material modelling method is suggested which proved capable of handling elastic-perfectly plastic material to multi-linear elastic – muti-linear elastic-plastic material. No additional change in coding is required in handling such material characteristics. The stress-strain relation of a material can be discretised into multiple straight lines, may it be bi-linear or multi-linear in elastic or elastic-plastic zone. Hysteresis and Bauschinger (kinematic hardening) effects were included in the modelling algorithms. Cowper and Symonds model of strain rate sensitivity was included.

### 7.2.8 INSTANTANEOUS NUMERICAL VALUE AND EFFECTIVE VALUE

Instantaneous velocity and acceleration obtained from the processor's central difference temporal integration scheme cannot be used as the effective velocity and acceleration. This has been demonstrated with a numerical example, where the instantaneous velocity was fluctuating and the acceleration was oscillating with much higher amplitude and frequency, than physical limits. These have been clearly shown and concluded that both the instantaneous velocity and acceleration is not representing the real values.

A method based on statistical Least Squares Method is proposed for obtaining the effective velocity and acceleration. The method is capable of predicting statistically sound effective velocity and acceleration. This has been proven a good method with successful predictions in numerous numerical examples presented in the thesis.

### 7.2.9 CONTACT-IMPACT ALGORITHMS

Both the FDM and the MSS provide contact-impact treatment. The FDM only handles contact-impact between the structure and a static rigid cylindrical obstacle. While the MSS can handle almost every contact-impact scenario.

The FDM contact element definition is simple, only one contact node is required. The FDM provides a more versatile contact search compared to the MSS contact-impact algorithm. The contact search method was successfully implemented and the contact searching outcome is satisfactory.

The MSS contact element uses the common node-surface definition. The contact searching is based on the defined contact elements. When comparing to the FDM contact searching, the MSS contact searching is more primitive. This is owing to the nature of the MSS contact element structure. The process to define the contact element is relatively tedious. Additional algorithms may assist the user from the tedious element definition task as well as a better contact searching, similar to the FDM.

In contact searching, the FDM does not consider potential voids and overlaps. The MSS incorporates a novel algorithm for handling potential void and overlap. Different to the conventional method, the proposed method provides better implementation and contact searching of node-node and node-surface contacts.

Two sets of contact releasing conditions are available with the proposed contact algorithms. Both releasing conditions are good with some contact models but not all. Anyhow, the two sets of contact releasing conditions are capable to cover all the proposed contact treatments.

Two novel treatments in predicting the normal response of the contact element were proposed. They are Adaptive Penalty method (AP) and Kinematic Contact Algorithm (KCA). Another method with combining both the AP and the KCA has proved to be a successful approach. It is called KCA/AP method. All the above three methods are capable in predicting the normal contact response of a contact element. From the study of all the methods including the well-known Penalty Method, the KCA/AP method gives the best in terms of the results. The MSS contact-impact algorithms incorporated Coulomb's friction model in predicting the tangent contact-impact response.

Integration of all contact elements, especially neighbouring elements, creates additional energy using direct superimposition method. A method called Energy Conservation

Superimposition method is suggested. The method has successfully controlled the energy level of integrating all the contact elements.

### 7.2.10 REVOLUTE JOINT ELEMENT

A novel revolute joint element is proposed and implemented in the MSS. The element is capable of handling resistive moment, rotary limits, locking, and rotary damping. The arrangement of the revolute joint element is a better solution to the ANSYS COMBIN-7 element.

The new element provides a better way of defining the resistive torque, which is in variations, when comparing to the ANSYS element. A novel and adaptive method in predicting the rotary limit response is suggested, which is capable of predicting the rotary stiffness when reaching rotary limits. Conventional Penalty-like method requires pre-defined user parameters, which is a definite drawback of the method. Locking mechanism is included, which is not available in the ANSYS element. A series of method in predicting the rotary response after reaching the rotary limits is also developed. They are Momentum-Balanced method, Forced-Displacement method, and Energy Conservation method. By and large, the Adaptive Stiffness method is a better method to be used, both in terms of implementation efficiency and the predicted response.

Preliminary study of the element stability is carried out and the element is conditional stable especially involving locking mechanism. A stability criterion for the revolute element is proposed. Nothing is mentioned by the literature about the stability of the revolute element and others of its kind.

### 7.2.11 HUMAN BODY MOTION STUDIES

Simulations are carried out for human body response subjected to frontal impacts. The capability of the MSS in modelling human body subjected to collision is shown. The development of an idealised human body was detailed. A $2^4$ factorial design of experiment is used to study the effect of human height, human weight, collision speed, and obstacle height towards human head injury. The study showed that, the collision speed, the location of obstacle and combined effect of both factors contribute the most towards linear acceleration at the head. From this, it is concluded that the design of the vehicle such as motorcycle should consider the location of first possible contact during a collision, as it will significantly influence the risk of head injury.

An idealised motorcycle model based on the Malaysian National Motorcycle, KRISS 110, was developed. The model is just good enough serving as an initial development. An idealised human model was placed onto the motorcycle model and simulation was conducted. From the development of the motorcycle to the solution of the simulation, one can conclude that further works are required in the development of the motorcycle model. The collision effect (impact mode) from the wheel was ignored, the structure bending strength at the joints and the mass distribution were assumed. To overcome these problems, technical data is required from the manufacturer and tests are needed, especially in finding the impact response of certain parts of the motorcycle, like the front fork and the wheel. The simulation solution showed the contact of human leg contact with the motorcycle was not handled considerably. More detailed model of the motorcycle is required to achieve such contact. The contact-impact interface between the human legs and the motorcycle structures, such as the front basket, cowl, etc, has to be considered. Furthermore, the $2^K$ factorial design result showed that the location of the initial collision location significantly influences the post-impact acceleration or/and deceleration of the occupant.

### 7.2.12 DESIGN OF SLED TEST RIG

A Conceptual design of the sled test rig is illustrated. From the available model as well as the conceptual design, the set up of the facility requires a large amount of money, easily more than 250,000 pounds. Large portion of this money would be needed to buy the high-speed image capturing facilities and the human dummy. In addition, the sled test facility needs a large area to accommodate.

## 7.3 CONCLUSIONS

- Initially a numerical technique based simulation system, FDM, has been developed based on numerical models from the literature. The development incorporated the conventional programming techniques. The system includes its own pre- and post-processors.

- Subsequently, an object-oriented designed numerical-based simulation system is developed, namely the MSS, which is more efficient than the FDM. The Object-oriented design was used in the MSS pre- and post- processors development as well. Many novel elements are incorporated in the MSS. The MSS is capable to model most of impact numerical problems of 1D structure in 2D space.

- Two novel 1D structural elements were designed and developed by the author, namely Bending-Division and Node-Layer elements. Validations were made.

- A distinctive method for modelling material property has been illustrated. The idealised the stress-strain relation in all sort of situation from elastic - perfectly plastic to multi-linear elastic – multi-linear elastic-plastic.

- Contact-impact searching algorithms have been developed, and a novel treatment of any potential void and overlap has been suggested. The method further differentiated node-node contact and node-surface contact.

- Four novel treatments of normal contact response have been developed. One was implemented to handle contact with static rigid bar-like obstacle. The other three are KCA, AP and KCA/AP. Different releasing conditions were studied along with the methods. Comparisons and validations were made among the methods, the conventional Penalty method, and exact solutions.

- Coulomb's friction model has been incorporated in the contact elements of MSS (with KCA, AP or KCA/AP). Validations with theoretical result have been made.

- Energy Conservation Superimposition method has been proposed to integrate contact elements. Direct superimposition will generate extra energy, which is unacceptable.

- A novel revolute element has been proposed which is capable of handling rotary limits, locking, resistive moments, and viscous damping.

- Three numerical methods in predicting resistive torque was designed and incorporated into the revolute joint element. They are Adaptive Stiffness, Momentum-balanced, and Energy Conservation methods. Comparisons and validations were carried among the above methods, the conventional Forced-Displacement method, the ANSYS COMBIN7 Penalty-like method, experimental result, and exact solution.

- Preliminary study of the instability of the revolute joint was conducted and a stability criterion was suggested.

- Idealised human model was developed based on the information from the literatures.

- Simulation studies have been carried out based on a $2^4$ factorial design of experiments. Most significant effects were found. The outcome of the studies can be used as a guideline in design of the motorcycle, where the obstacle of the first contact should be at the lower end of the lower extremities. The better understanding of human height and weight affecting the impact severity can be achieved.

- Malaysian's KRISS 110 motorcycle has been idealised as a simple model and a human model has been successfully incorporated. Simulations showed the success of such combined-model and the possibility of the MSS as a tool to simulate problems of similar kind.

- Conceptual design of a sled test rig has been carried out.

## 7.4 RECOMMENDATIONS FOR FURTHER WORK

The recommendations proposed here are intended as guidelines for future expansion of the simulation system as well as development of new elements. The recommendations also include further steps in the human body motion studies subject to frontal collisions, especially riding a motorcycle.

### 7.4.1 DEVELOPMENT OF SIMULATION SYSTEM

Before developing future generation of similar numerical method based simulation system, the element design rules and guidelines have to be standardised, in object-oriented way. Future development as well as incorporation of any existing elements into the system will have to follow this scheme. The feasibility has been proved by incorporating several existing well-known elements together into the MSS. With standardised guidelines, the next generation of simulation system can then be developed.

The MSS pre-processor user interfaces need to be upgraded. The present pre-processor is still lacking of user-friendliness. Furthermore, some modelling tasks are tedious and repetitive. Future versions should eliminate these shortcomings.

The MSS processor is at its peak shape, except lacking of some useful features. The user should be able to stop the simulation temporarily and view the result in the post-processor then continue the processing if necessary. Besides that, the instantaneous energy levels should be shown.

The MSS post-processor has several imperfections, which need extra attentions. The contour plotting and graph plotting within the post-processor are not available. Before going further for incorporating these features, attention has to be paid in dealing with 2D and 3D elements in 3D space.

### 7.4.2 STRUCTURE ELEMENTS

Other more complicated elements, such as 1D link element in 3D space, 2D plane element in 2D space, 2D shell element in 3D space, etc, are needed. Thus development of more versatile elements is the next steps in the structure element design and development. During the development, the rules of object-oriented design have to be followed.

### 7.4.3 NUMERICAL STABILITY

Further numerical stability studies have to be conducted in order to solve the problem completely.

### 7.4.4 MATERIAL MODELS

Other material models such as failure model for fracture studies, crushable foam and rubber models can be incorporated into the MSS. These models will dramatically enhance the potential of the simulation system.

### 7.4.5 CONTACT-IMPACT ALGORITHMS

More versatile contact-impact algorithms are required. They must be able to automate the element formations with better contact searching in terms of efficiency and speed. This is the main drawback of the present contact-impact algorithms.

### 7.4.6 STABILITY OF THE REVOLUTE JOINT ELEMENT

More studies have to be carried out to understand the stability of the revolute joint. The present work is just the preliminary approach. Further validations are required. The cause of the instability needs to be established and verified.

### 7.4.7 HUMAN MODEL AND MOTION STUDIES

More detailed simulations can be carried out, such as $3^k$ factorial design. Human model with muscular responses during the critical situation can be incorporated in the future, by having amended Revolute Joint elements. A more detailed model will have to be constructed, such as considering the real centre of gravity of each human segment.

### 7.4.8 HUMAN-MOTORCYCLE MODEL

The motorcycle model needs to be refined. More geometrical details are required. This will enable better modelling of the human-motorcycle contacts. Furthermore, the bending strength between the connected components is required.

# REFERENCES

1. E.A. Witmer, H.A. Balmer, J.W. Leech and T.H.H. Pian, Large dynamic deformations of beams, rings, plates, and shell, *AIAA Journal*, **1**(8), 1848-1857 (1963).

2. M.J. Fagan, *Finite Element Analysis: Theory and Practice*, Longman Scientific & Technical, (1992).

3. J.W. Thomas, *Numerical Partial Differential Equations: Finite Difference Methoads*, Springer-Verlag (1995)

4. L. Karaoglan and A.K. Noor, Assessment of temporal integration schemes for the sensitivity analysis of frictional contact/impact response of axisymmetric composite structures, *Comput. Methods Appl. Mech. Engrg.*, **130**, 369-393 (1996).

5. M.L. Wilkins, R.E. Blum, E. Cronshagen and P. Grantham, *A Method for Computer Simulation of Problems in Solid Mechanics and Gas Dydnamics in Three Dimensions and Time*, Lawrence Livermore Laboratory, Report UCRL-51574, Rev. 1, (1975).

6. B. Alder, S. Fernbach and M. Rotenberg, *Mehtods in Computational Physics*, Vol. 3, Academic Press, New York, (1963).

7. J.A. Zukas, T. Nicholas, H.F. Swift, L.B. Greszczuk, D.R. Curran, *Impact Dynamics*, John Wiley & Sons, Inc., (1982).

8. J.O. Hallquist, *LS-DYNA3D User Manual*, Livermore Software Technology, Livermore, (1991).

9. F.M. Pappas and E. Thro, *The Visual C++ Handbook, 2nd Edition*, Osborne McGraw-Hill, (1995).

10. K. Gregory, *Special Edition: Using Visual C++ 5*, Que, (1997).

11. Microsoft Corporation, Microsoft Foundation Class Library Reference, Microsoft Press, (1995).

12. F. Pandolfi, M. Oliver and M. Wolski, *Microsoft Foundation Class 4 Bibble, Waite Group Press*, (1996).

13. D.J. Benson, Computational methods in Lagrangian and Eulerian hydrocodes, *Computer Methods in Applied Mechanics and Engineering*, 235-394 (1992).

14. T.B. Belytschko, J.M. Kennedy and D.F. Schoeberle, On finite element and difference formulations of transient fluid-structure problems, in *Proc. Computational Methods in Nuclear Engineering*, Charleston, SC, Amer, Nuclear Society, IV39-IV54 (1975).

15. J.W. Leech, T.H.H. Pian, E.A. Witmer and W. Herrmann, Dynamic response of shells to externally applied dynamic loads, *Aeronaut Systems Div TDR-62-610*, (1962).

16. K.J. Bathe, E. Ramm and E.L. Wilson, Finite element formulations for large deformation dynamic analysis, *Int. J. Numerical Methods in Engng*, **9**, 353-386 (1975).

17. D.W. Murray and E.L. Wilson, Finite element large deflection analysis of plates, *J. Engng Mech. Div., ASCE,* **94**, 143-165 (1965).

18. C.A. Flippa, Refined finite element analysis of linear and nonlinear two-dimensional structures, *SESM Report,* No. 66-22, Dept. Civ. Engng, Univ. of California, Berkeley (1966).

19. S. Yaghamai, Incremental analysis of large deformations in mechanics of solids with applications to axisymmetric shells of revolution, *SESM Report*, No. 69-17, Dept. of Civ Engng, Univ. of California, Berkeley (1968).

20. P. Sharifi and E.P. Popov, Nonlinear buckling analysis of sandwich arches, *J. Engng Mech. Div., ASCE,* **97**, 1397-1412 (1970).

21. S. Yaghmai, Incremental analysis of large deflections of shells of revolution, *Int. J. Solids Struct.*, **7**, 1375-1393 (1971).

22. P.V. Marcal, The effect of initial displacements on problems of large deflection and stability, *Tech. Report,* ARPA E54, Brown University, Division of Engineering (1967).

23. H.D. Hibbitt, P.V. Marcal and J.R. Rice, Finite element formulation for problems of large strain and large displacements, *int. j. Solids Struct.*, **6**, 1069-1086 (1970).

24. P. Sharifi and D.N. Yates, Nonlinear tehermo-elastic-plastic and creep analysis by finite element method, *AIAA paper,* No 73-358, *AIAA/ASME/SAE 14th Structures, Struct. Dyn. Materials Conf.,* Williamsburg, Virginia (1973).

25. Y. Yamada, Incremental formulation for problems with geometric and material nonlinearities, *Advances in Computational Methods in Structure Mechanics and Design*, 2nd US-Japan Seminar matrix Meth. Struct. Analysis Design, Univ. of Alabama Press, 325-355 (1972).

26. J.A. Stricklin, W.A. Von Riesemann, j.R. Tillerson and W.E. Haisler, Static geometric and material nonlinear analysis, *Advances in Computational Methods in Structural mechanics and Design*, 2nd US-Japan Seminar matrix Meth. Struct. Analysis Design, Univ. of Alabama Press, 301-324 (1972)

27. G.A. Dupuis, H.D. Hibbit, S.F. McNamara and P.V. Marcal, Nonlinear material and geometric behavior of shell structures, *Computers Struct.*, **1**. 223-239 (1971).

28. S.J. Hashmi, S.T.S. Al-Hassani and W. Johnson, Large deflection elastic-plastic response of certain structures to impulsive load: numerical solutions and experimental results, *Int. J. Mech. Sci.*, **14**, 843-860 (1972).

29. W.F. Noh, CEL: a time-dependent, two-space-dimensional, coupled Eulerian-Lagrange code, in: *Methods in Computational Physics, Vol 3, Fundamental Methods in Hydrodynamics*, Academic Press, New York, 117-179 (1964).

30. M. Wilkins, Calculation of elastic-plastic flow, in: *Methods in Computational Physics, Vol 3, Fundamental Methods in Hydrodynamics*, Academic Press, New York, 211-263 (1964).

31. S. Hancock, PISCES SDELK Theoretical Manual, Physics International, (1985).

32. R.B. Demuth, L.G. Margolin, B.D. Nichols, T.F. Adams and B.W. Smith, *SHALE: A computer program for solid dynamics*, Los Alamos National Laboratory, LA-10236, (1985).

33. A.A. Amdsden, H.M. Ruppel and C.W. Hirt, *SALE: A simplified ALE computer program for fluid flow at all speeds*, Los Alamos Scientific Laboratory, (1980).

34. G. Maenchen and S.Sack, the TENSOR Code, in: *Methods in computational Physics*, Vol. 3, Academic Press New York, (1964).

35. A.A. Amsden and C.W. Hirt, YAQUI: *An arbitary Lagrangian-Eulerian computer program for fluid flow at all speeds*, Los Alamos Scientific Laboratory, LA-5100, (1973).

36. S. Vasudevan, H. .Okada, S.N. Atluri, Development of new spatially curved frame finite element for crash analysis, part 1: formulation and validation, *Computational Mechanics*, **16**, 426-436 (1995).

37. M. Iura and S.N. Atluri, On a consistent theory, variational formulation of finitely stretched and rotated 3-D space-curved beams, *Computational Mechanics,* **4**, 73-88 (1989).

38. S. Vasudevan, H. .Okada, S.N. Atluri, Development of a new frame finite element for crash analysis, using a mixed variational principle and rotations as independent variables, *Finite Elements in Analysis and Design*, **23**, 155-171 (1996).

39. T.J.R. Hughes, M. Cohen and M. Haroun, Reduced and selective integration techniques in the finite element analysis of plates, *Nuclear Engineering and Design*, **46**, 203-222 (1978).

40. E.S. Mistakidis, C.C. Baniotopoulos and P.D. Panagiotopoulos, An effective two-dimensional numerical method for the analysis of a class of steel connections, *Computational Mechanics*, **21**, 363-371 (1998).

41. R.L. Woodward, Modeling geometrical and dimensional aspects of ballistic penetration of thick metal targets, *Int. J. Impact Engng.*, **18**(4), 369-381(1996).

42. M.S.J. Hashmi and P.J. Thompson, Numerical method of analysis for the mushrooming of flat ended projectiles impacting on a flat rigid anvil, *Int. J. Mech Sci.,* **19**(5), 273-283 (1977).

43. H. Abbas, D.K. Paul, P.N. Godbole and G.C. Nayak, Soft missle impact on rigid targets, *Int J. Impact Engng*, **16**(5/6), 727-737 (1995).

44. Y.M. Chen, Numerical computation of dynamic stress intensity factors by a Lagrangian finite difference method (the HEMP codes), *Engng Fracture Mech.*, **7**, 653-660 (1975).

45. J.A. Aberson, J.M. Anderson and W.W. King, Dynamic analysis of cracked structures using singularity finite elements, in *Mechanics of Fracture 4, Elastodynamic Crack problems,* G.C. Sih, ed., Noordhoff International , Leyden, 263 (1977).

46. B. Brickstad, A FEM analysis of crack arrest experiments, *Int. J. Fracture*, **21**, 177-194 (1983).

47. A.S.M. Israil and G.F. Dargush, Dynamic fracture mechanics studies by time-domain BEM, *Engng Fracture Mech.*, **39**, 315-328 (1991).

48. X. Lin and J. Ballmann, Re-consideration of Chen's problem by finite difference method, *Engng Fracture Mech.*, **44**, 735-739 (1993).

49. Y.G. Zhang and J. Ballmann, An explicit finite difference procedure for contact-impact analysis of crack edges, *Archive of Applied Mechanics*, **66**, 493-502 (1996).

50. T.J. Clifton, A difference method for plane problems in dynamic elasticity, *Q. Appl. Math.*, **25**(1), 97-116 (1967).

51. W.W. Recker, A numerical solution of three-dimensional problems in dynamic elasticity, *Trans. ASME J. Apppl. Mech.*, **37**, 116-122 (1970).

52. K. Liu and T. Yokoyama, Dynamic behavior of elastic/viscoplastic bars of squarecross section subjected to longitudinal impact, *Trans. JSME*, **58**(547)A, 449-456 (1992).

53. K. Liu and S. Tanimura, Numerical analysis for dynamic stress concentration in a rectangular block due to impact, *Int. J. Impact Engng*, **19**(8) 653-666 (1997).

54. K. Kaizu, S. Tokunaga and S. Tanimura, Dynamic behavior of thick plates due to oblique impact (numerical analysis of three-dimensional stress waves in an elastic/viscoplastic body), *JSME Int. J. Series A*, **39**(1), 78-85 (1996).

55. F.L. Huang and Z.P. Wang, Modeling fracture process in a ductile solid under intense impulsive loading using a model of the void growth with temperature-dependency, *Engng Fracture Mech.*, **55**(4), 657-674 (1996).

56. R. Davalos and B. Rubinsky, An evolutionary-genetic approach to heat transfer analysis, *J. Heat Transf. Tans. ASME*, **118**(3), 528-532 (1996).

57. B. Rubinsky and R. Davalos, The use of evolutionary-genetic analogy in numerical analysis, *Communications in Numerical Methods in Engineering*, **14**, 151-160 (1998).

58. O.P. Agrawal, K.J. Danhof and R. Kumar, A superelement model based parallel algorithm for vehicle dynamics, *Computers & Stuctures*, **51**(4), 411-423 (1994).

59. D.B. Bathe, J.R. Baumgardner, J.H. Cerutti, B.J. Daly, K.S. Holian, E.M. Kober, S.J. Mosso, J.W. Painter, R.D. Smith and M.D. Torrey, PAGOSA: a massively-parellel, multi-material hydrodynamics model for three-dimensional high-speed flow, and high-rate material ideformation, in: *Proc. 1993 SCS Simulation Multiconference: High Performance Computing*, (29 March – 1 April 1993), 9-14 (1993).

60. K.D. Kimsey and M.A. Olson, Parallel computation of impact dynamcs, *Comput. Methods Appl. Mech. Engng.*, **119**, 113-121 (1994).

61. U.N. Gandhi and S.J. Hu, Data-based approach in modeling automobile crash, *Int. J. Impact Engng*, **16**(1), 95-118 (1995).

62. C. Lage, The application of object-oriented methods to boundary elements, *Comput. Methods Appl. Mech. Engng.*, **157**, 205-213 (1998).

63. W. Hackbusch and Z.P. Nowak, On the fast matrix multiplication in the boundary element method by panel clustering, *Numer. Math.*, **54**94), 463-491 (1989).

64. R.L. Woodward and J.P. Lambert, A discussion of the calculation of forces in the one-dimensional finite difference model of Hashmi and Thompson, *Int. J. Mech Sci.*, **23**(8), 497-501 (1981).

65. M.S.J. Hashmi and F.B. Klemz, The effect of friction in simple upsetting of cylindrical billets of elastic-plastic and elastic-strain hardening material: A numerical technique, Journal of Nuclear Materials, **3**, 597-604(1976).

66. M.S.J. Hashmi and F.B. Klemz, Closed die forging of axisymmetric shapes from cylindrical billets of strain hardening and strain rate sensitive material: Experimental results and theoretical predictions, in *Proceedings: 19th int. MTDR conf.*, 61 (1978).

67. A.M.S. Hamouda, M.S.J. Hashmi, Simulation of the impact of a tool steel projectile into copper, mild-steel and stainless-steel (304) test specimen, in *Proceedings: the 2nd International Conference on Structures Under Shock and Impact II*, Jun 11-18, 1992, 52-61(1992).

68. A.M.S. Hamouda, S. Sulaiman, M.S.J. Hashmi, Fast upsetting of circular cylinders of aluminium metal matrix composites: experimental results and numerical analysis, *Jounal of Material Processing Technology*, **60**(1-4), 723-727 (1996).

69. J. Argyris, H. Balmer and I. St. Doltsinis, Some thoughts on shell modelling for crash analysis, *Computer Methods in Applied Mechanics and Engineering*, **71**, 341-365 (1988).

70. J. Argyris, H.A. Balmer, J. St. Doltisinis, On shell models for impact analysis, in: *Analytical and Computational Models of Shell*, CED-Vol. 3, A.K. Noor *et al.*, eds., ASME Winter Annual Meeting (ASME, San Francisco, Dec 1989), 443-456 (1989).

71. J. Argyris, H.A. Balmer, J. St. Doltisinis, Natural finite differences in shell modelling for impact analysis, in: *Discretisation Methods in Structural Mechanics*, G. Kuhn and H. Mang, eds., IUTAM/IACM Symposium, Vienna (June 1989) 133-143 (1989).

72. J. Argyris, H.A. Balmer, J. St. Doltisinis and A. Kurz, Computer simulation of crash phenomena, *Int. J. Numer. Meths. Engrg.*, **22**, 497-519 (1986).

73. J. Argyris, H.A. Balmer, J. St. Doltisinis, A simple but subtle model for the analysis of shell-like structures, *Computer Methods in Applied Mechanics and Engineering*, **85**, 1-20 (1991).

74. I. Newton, *Philosophiae Naturalis Principia Mathematica,* Royal Society, London, (1687).

75. L. Euler, Nova methods motum corporum rigidarum determinandi, *Novi Commentarii Academiao Scientiarum Petropolitanae*, **20**, 208-238 (1776).

76. R. Schwerassek, R.E. Roberson, A perspective on computer-oriented multibody dynamical formalisms and their implementations, in *Dynamics of Multibody Systems*, G. Bianchi and W. Schiehlen, eds., Springer-Verlag, Berlin, 263-273 (1986).

77. W. Schiehlen, Multibody system dynamics: roots and perspectives, *Multibody System Dynamics,* **1**, 149-188 (1997).

78. M. Otterm M. Hocke, A. Daberkow and G. Leister, An object-oriented data model for multibody systems, in *Advanced Multibody System Dynamics*, W. Schiehlen (ed.), Kluwer Academic Publishers, Dordrecht, 19-48 (1993).

79. P. Nikravesh, I. Chung and R.L. Benedict, Plastic hinge approach to vehicle crash simulation, *Computers & Structures*, **16**(1-4), 395-400 (1983).

80. J. Ambrosio, P. Nikravesh and M.S. Pereira, Crashworthiness analysis of a truck, *Journal of Mathematical Computer Modelling*, **14**, 959-964 (1990).

81. J.P. Dias and M.S. Pereira, Design for vehicle crashworthiness using multibody dynamics, *Int. J. Vehicle Design*, **15**(6), 563-577 (1994).

82. A.Cardona and M. Geradin, A beam finite element nonlinear theory with finite rotations. *Int . J. Numer. Meth. Engng*, **26**, 2403-2438 (1988).

83. J.A.C. Ambrosio, Dynamics of structures undergoing gross motion and nonlinear deformations: a multibody approach, *Computers & Structures*, **59**(6), 1001-1012 (1996).

84. G. Yang and A.D. de Peter, The determination of the nonlinear motion of a railway vehicle, in: *The dynamics of Vehicles: on Roads and on Tracks, Proc. 12th IAVSD-Symposium*, Lyon, France, (26 August- 30 1991), 225-239 (1991).

85. M.S. Pereira, J.A.C. Ambrosio and J.P. Dias, Crashworthiness analysis and design using rigid-flexible multibody dynamics with application to train vehicles, *Int. J. Numerical Methods in Engng*, **40**, 655-687 (1997).

86. D. Ma and H. Lankarani, A multibody/finite element analysis approach for modeling of crash dynamic responses, *Advances in Design Automation*, **1**, DE-Vol. 69-1, ASME, 55-65 (1994).

87. D. Ma and H. Lankarani, A multibody/finite element analysis approach for modeling of crash dynamic responses, *Trans. ASME: J. Mechanical Design*, **119**, 382-387 (1997).

88. J.N. Lee and P.E. Nikravesh, Steady-state analysis of multibody systems with reference to vehicle dynamics, *Nonlinear Dynamics*, **5**, 181-191 (1994).

89. W.J. Stronge, Rigid body collisions with friction, *Proc. R. Soc. London.* A **431**, 169-181 (1990).

90. K.L. Johnson, *Contact Mechanics*, Cambridge University Press, Cambridge, (1985).

91. G. Eason, The displacement produced in an elastic half-space by a suddenly applied surface force, *Journal of the Institute of Mathematics and its Applications*, **2**, 299-326 (1966).

92. S.C. Hunter, Energy absorbed by elastic-waves during impact, *Journal of mechanics and Physicsof Solids*, **5**, 162.

93. C.W. Kilmister, and J.E. Reeve, *Rational Mechanics*, Longman, London, (1966).

94. Z-H Zhong, *Finite Element Procedure for Contact-Impact Problems*, Oxford Universities Press, Oxford, (1993).

95. B. Paul and J. Hashemi, An improved numerical method and computer program for counterformal contact stress problems, *Computer Techniques for Interface Problems (Editors: K.C. Park and D.K. Gartling)*, ASME, NY, 165-180 (1978).

96. T.J.R. Hughes, R.L. Taylor, J.L. Sackman, A. Curnier and W. Kanoknukulchai, A fintie element method for a class of contact-impact problems, *Computer Methods in Applied Mechanics and Engineering*, **8**, 249-276 (1976).

97. J.O. Hallquist, A numerical treatment of sliding interfaces and impacts, *Computer Techniques for Interface Problems (Editors: K.C. Park and D.K. Gartling)*, ASME, NY, 165-180 (1978).

98. A.B. Chaudhary and K.J. Bathe, A solution method for static and dynamic analysis of 3-D contact problems with friction, *Computer and Structures*, **24**(6), 855-873 (1986).

99. T. Belytschko and M.O. Neal, Contact-impact by the pinball algorithm with penalty and Lagrangian methods, *International Journal for Numerical Methods in Engineering*, **31**, 547-572 (1991).

100. Y.G. Zhang and J. Ballmann, An explicit finite difference procedure for contact-impact of crack edges, *Archieve of Apllied Mechanics*, **66**, 493-502 (1996).

101. J.O. Halliquist, *Nike 2-D – a vectorized, implicit, finite deformation, finite element code for analyzing the static and dynamic response of 2D solids*, RPT. UCRL-52678, LLNL, Livermore, CA, (1979).

102. J.O. Hallquist, *LS-DYNA3D Theoretical Manual*, Livermore Software Technology, Livermore, (1991).

103. D.J. Benson and J.O. Hallquist, *A Single Surface Contact Algorithm for the Post-buckling Analysis of Shell Structures*, University of California, San Diego, La Jolla, (1987).

104. P. Kohnke, *ANSYS User's Manual: Volume IV (Theory), Revision 5.1*, Swanson Analysis System, inc., Houston, PA, (1994).

105. Z.H. Zhong and J. Mackerle, Contact-impact problems: a review with bibiography, *Mechanics of contact impact*, editor: Miloslav Okrouhlik, Appl mech Rev **47**(2), ASME, 55-76(1994).

106. R.G. Whirley and B.E. Engelmann, Automatic contact in DYNA3D for vehicle crashworthiness, in: *Crashworthiness and Occupant Protection in Transportation System*, AMD-**169**/BED-**25**, ASME, 15-29 (1993).

107. J.R. Hughes, R.L. Taylor, J.L. Sackman, A. Curnier and W. Kanoknukulchai, A finite element method for a class of contact-impact problems. *Comput. Meth. Appl. Mech. Engng.*, **8**, 249-276 (1976).

108. K.J. Bathe and A. B. Chaudhary, A solution method for planar and axisymmetric contact problems, *Int. J. Numer. Meth. Engng*, **21**, 65-88 (1985).

109. N. Kikuchi and Y.J. Song, Remarks on relations between penalty and mixed finite element methods for a class of variational inequalitites, *Int. J. Numer. Meth. Engng.*, **15**, 1557-1561 (1980).

110. J.T. Oden, Exterior penalty methods for contact problems in elasticity, in: *Nonlinear Finite Element Analysis in Structural Mechanics*, W. Wunderlichm, E.Stein and K.J. Bathe, eds., Springer, New York, (1981).

111. J.O. Hallquist, G.L. Goudreau and D.J. Benson, Sliding interfaces with contact-impact in large-scale Lagrangian computations, *Int. J. Numer. Meth. Engng.*, **18**, 343-350 (1982).

112. T. Belytschko and M.O. Neal, Contact-impact by the pinball algorithm with penalty and Lagrangian methods, *International Journal for Numerical Methods in Engineering*, **31**, 547-572 (1991).

113. L. Jiang and R.J. Roggers, Combined Lagrangian multiplier and penalty function finite element technique for elastic impact analysis, *Computers & Structures*, **30**(6), 1219-1229 (1988).

114. H.M. de la Fuente and C.A. Felippa, Ephemeral penalty functions for contact-impact dynamics, *Finite Elements in Analysis and Design*, **9**, 177-191 (1991).

115. K. Kawachi, H. Suzuki and F. Kimura, Simulation of rigid body motion with impulsive friction force, in *Proceedings of the 1997 IEEE International Symposium on Assembly and Task Planning*, Marina del Rey, CA, August 1997, 182-187 (1997).

116. J.C. Simo, P. Wriggers and R.L. Taylor, A perturbed Lagrangian formulation for the finite element solution of contact problems, *Comp. Meth. Appl. Mech. Engng.*, **50**, 163-180 (1985).

117. P. Wriggers and J.C. Simo, A note on tangent stiffness for fully nonlinear contact problems, *Commun. Appl. Numer. Meth.*, **1**, 199-203 (1985).

118. B. Nour-Omid and P. Wriggers, A two-level iteration method for solution of contact problems, *Comput. Method. Appl. Mech. Engng.*, **54**, 131-144 (1986).

119. R.L. Taylor, Crash dynamics: methods for contact constraints in implicit and explicit transient solutions, in: *Crashworthiness and Occupant Protection in Transportation System,* AMD-**169**/BED-**25**, ASME, 1-14 (1993).

120. A.L. Eterovic and K.J. Bathe, On the treatment of inequality constraints arising from contact conditions in finite element analysis, *Comput. & Struct.*, **40**(2), 203-209 (1991).

121. P. Wriggers, J.C. Simo and R.L. Taylor, Penalty and augmented Lagrangian formulations for contact problams, in: *Proc. Int. Conf. Numer. Meth. Engng. – Theory and Applications (NUMETA'85),* J. Middleton and G.N. Pande, eds., 97-106 (1985).

122. J.B. Bohm, A comparison of different contact algorithms with applications, *Comput. Struct.*, **26**, 207 (1987).

123. L.M. Taylor and D.P. Flanagan, *PRONTO3D A 3-D Transient Solid Dynamics Program*, Sandia Report: SAND87-1912, UC-32, (1989).

124. N. Kikuchi and J.T. Oden *Contact Problems in Elasticity: A Study of Variational Inequalities and Finite Element Methods*, SIAM, Elsevier, Amsterdam, (1988).

125. M.H. Refaat and S.A. Meguid, On the elastic solution of frictional contact problems using variational inequalities, *Int. J. Mech. Sc.*, **36**(4), 329-342 (1994).

126. M.H. Refaat and S.A. Meguid, On the contact stress analysis of spur gears using variational inequalities, *Computers & Structures*, **37**(3), 871-882 (1995).

127. M.H. Refaat and S.A. Meguid, A novel finite element approach to frictional contact problems, *Int. J. for Numerical Methods in Engineering*, **39**, 3889-3902 (1996).

128. J.R. Baber, Adhesive contact during the oblique impact of elastic spheres, *J. Appl. Mathematics and Physics*, **30**, 468-476 (1979).

129. J.T. Oden and J.A.C. Martins, Models and computational methods for dynamic friction phenomena, *Com Meth in Appl Mech and Eng*, **52**, 527-634 (1985)

130. B. Fridriksson, *On elastostatic contact problem with friction*, Dissertations No. 6, linkoping University, Linkoping, (1976).

131. A. Curnier, A theory of friction, *Int J Solids and Struct*, **20**(7), 637-647 (1984).

132. N. Kikuchi and J.T. Oden, *Contact Problems in Elasticity: A Study of Variational Inequalities and Finite Element Methods*, SAIM Publication, Philadelphia, (1988).

133. L. Karaoglan and A.K. Noor, Assessment of temporal integration schemes for the sensitivity analysis of frictional contact/impact response of axisymmetric composite structures, *Comput. Methods Appl. Mech. Engrg.*, **130**, 369-393 (1996).

134. D. Sha, K.K. Tamma and M. Li, Robust explicit computational developments and solution strategies for impact problems involving friction, *International Journal for Numerical Mehtods in Engnieering*, **30**, 721-739 (1996).

135. J.B. Keller, Impact with friction, *ASME J. Appl. Mech.*, **53**, 1-4 (1986).

136. R.M. Brach, Rigid body collisions, *ASME J. Appl. Mech.*, **56**, 133-138 (1989).

137. A.P. Ivanov, Energetic of a collision with friction, *J. Appl. Maths. Mechs.*, **56**(4), 527-534 (1992).

138. Y. Wang and M.T. Mason, Two-dimensional rigid body collisions with friction, *ASME Trans. J. Appl. Mech.*, **59**, 635-642 (1992).

139. E.J. Routh, *Dynamics of a System of Rigid Bodies: Elementary Part,* 7th Ed., Macmillan, London, 126-162 (1860).

140. P.E. Gauitier and W.L. Cleghorn, A spatially translating and rotating beam element for modeling flexible manipulators, *Mechanism and Machine Theory*, **27**, 415-433 (1992).

141. W.H. Sunada and S. Dubowsky, The application of finite element methods to the dynamics of flexible linkage systems, *ASME Journal of Mechanical Design*, **103**, 643-651 (1983).

142. R.J. Theodore and A. Ghosal, Comparison of assumed mode and finite element method for flexible multilink manipulators, *The International Journal of Robotics Research*, **14**, 91-111 (1995).

143. Y. Wnag and R.L. Huston, A lumped parameter method in the nonlinear analysis of flexible multibody systems, *Computers and Structures*, **50**, 421-432 (1994).

144. Y.A. Khulief and A.A. Shabana, Dynamic analysis of a constrained system of rigid and flexible bodies with intermittent motion, *ASME Journal of Mechanism, Transmission and Automation in Design* ,**108**, 38-45 (1986).

145. J. Rismantab-sany and A.A. Shabana, On the use of momentum balance in the impact analysis of constrained elastic systems, *ASME Journal of Vibration and Acoustics*, **112**, 119-126 (1990)

146. A.S. Yigit, A.G. Ulsoy and R.A. Scoot, Spring-dashpot models for the dynamics of a dynamics of a radially rotating beam with impact, *Journal of Sound and Vibration*, **142**, 515-525 (1990).

147. B.P. Nagaraj, B.S. Nataraju and A. Ghosal, Dynamics of a two-link flexible system undergoing locking: mathematical modeling and comparison with experiments, *Journal of Sound and Vibration*, **207**, 567-589 (1997).

148. S. Plagenhoef, F.G. Evans and T. Abdelnour, Anatomical data for analysis human motion, *Research Quaterly for Exercise and Sport*, **54**(2), 169-178 (1983).

149. J.G. Hay and J.G. Reid, *Anatomy, Mechanics, and Human Motion: 2nd ed.*, Prentice Hall, New Jersey, (1987).

150. O. Perey, Fractures of the vertebral end-plate in the lumbar spine: An experimental biomechanical investigation, *Acta Orthopaedica Scandinaviac*, **27,** 237 (1958).

151. S. Carlsoo, *How Man Moves: Kinesiological Methods and Studies*, Heinemann, London, (1972).

152. A.A. III White and M.M. Panjabi, The basic kinematics of the human spine, *Spine*, **3**, 16 (1978).

153. W.T. Dempster, Space requirements of the seated operator, *WADC Technical Report*, 55-159, Wright-Patterson Air Force Base, Dayton, OH, (1955).

154. W.M. Krogman and F.E. Johnson, Human mechanics: four monographs adridged, *WADC Technical Report*, AMRL-TDR-63-123, Wright-Patterson Air Force Base, Dayton, OH, (1963).

155. W.T. Dempster and G.R.L. Gaughran, Properties of body segments based on size and weight, *American J. Anatomy*, **120**, 33-54 (1965).

156. R. Contini, Body segment parameters, part ii, *Artificial Limbs*, **16**, 1-19 (1972).

157. R. Drills and R. Contini, Body segment parameters, *Technical Report*, 116.03, New York University School of Engineering and Science, (1966).

158. J. Santschi, J. DuBois and C. Omoto, Moments of intertia and centers of gravity of the living human body, *WADC Technical Report*, AMRL-TDR-63-36, Wright-Patterson Air Force Base, Dayton, OH, (1963).

159. C.E. Clauser, J.T. McConville and J.W. Young, Weight, volume and center of mass of segments of the human body, *WADC Technical Report*, AMRL-TDR-69-70, Wright-Patterson Air Force Base, Dayton, OH, (1969).

160. W. Braune and O. Fischer, Uber der Schwerpunkt des menschlichen Korpers mit Rucksucht auf die Ausrustung des Deutschen Infanteristen, *Abhandl. D. Kl. Sachs. Ges. Wiss. Math. Phys. Kl.*, **15**(7), 561-672 (1889).

161. D.A. Winter, *Biomechanics of Human Movement,* John Wiley & Sons, (1979).

162. P.Prasad and C.C. Chou, A review of mathematical occupant simulation models, *Crashworthiness & Occupant Protection in Transportation Systems*, AMD **106**/BED **13**, ASME, 95-112 (1989).

163. A.I. King and C.C Chou, Mathematical modeling, simulation and experimental testing of biomechanical system crash response, *J. Biomechanics*, **9**, 301-317 (1976).

164. P. Prasad, An overview of major occupant simulation models, *Mathematical Simulation of Occupant and Vehicle Kinematics*, SAE Publication, SAE Paper No. 840855 (1984).

165. J.A. Bartz, A three-dimensional computer simulation of a motor vehicle victim, Phase 1 – development of the computer program, *Calspan Report*, VJ-2978-V-1, (1971).

166. J. Wismans, J. Maltha, J.J. van Wijk and E.G. Janssen, MADYMO – a crash victim simulation computer program for biomechanical research and optimization of designs for impact injury prevention, AGARD meeting, Kooln, Germany, April (1982).

167. R.L. Huston, C.E. Passerello and M.W. Harlow, UCIN vehicle – occupant / crash vistim simulation model, *Structural Mechanics Software Series*, **II**, University Press of Virginia, 131-150 (1978).

168. R.G. Chandler and D.H. Laananen, Seat / occupant crash dynamic analysis verification test program, *SAE Paper*, 790590, (1979).

169. D. Ma, L.A. Obergefell and A.L. Rizer, Development of human articulating joint model parameters for crash dynamics simulations, *39th Stapp Car Crash Conference Proceedings*, Nov 8-10, Coronado, California, SAE 952726, 239-250 (1995).

170. L.A. Obergefell, T.R. Gardner, I. Kaleps and J.T. Fleck, *Articulated Total Body Model Enhancements, Volume 2: User Guide*, Armstrong Laboratory Technical Report, AAMRL-TR-88-043, WPAFB, Ohio, (1988).

171. A.D. Glanville and G. Kreezer, The maximum amplitude and velocity of the joint moments in normal male human adults, *Human Biology*, **9**, 197-211 (1937).

172. D.H. Laananen, *Computer Simulation of Aircraft Seat and Occupant(s) in a Crash Environment – Program SOMLA/SOM-TA User manual*, DOT/FAA/CT-90/4, Federal Aviation Administration Technical Center, Atlantic City Airport, New Jersey, may, (1991).

173. E.A. Engin, *Measurement of Resistive Torques in Major Human Jionts*, Report No. AAMRL-TR-79-4, WPAFB, Ohio, April (1979).

174. E.G. Engin and S.M. Chen, *Human Joint Articulation and Motion-Resistive Properties*, AAMRL-TR-87-011, WPAFB Ohio, (1987).

175. H.Cheng, L.A. Obergefell and A.L. Rizer, *Generator of Body Data (GEBOD) Manual*, Report No. AL/CF-TR-1994-0051, Amstrong Laboratory, WPAFB Ohio, (1994).

176. European Experimental Vehicles Committee (EEVC) ad-hoc Group, *Report on Motorcycle Safety*, Dec, (1993).

177. P.L. Harms, *Injury Patterns of Motorcyclists Involved in Accidents*, TRRL Supplementary Report SR651, Crowthrone TRL, UK, (1981).

178. J.J. Hurt *et al.*, *Motorcycle Accident Cause Factors and Identification of Counter-measures Vol.1: Technical Report*, Cont. No. DOT HS-5-01160, US Department of Transportation NHTSA, (1981).

179. P. Kalbe *et al.*, Trauma assessment of injuries and their consequences in accidents with two-wheelers, in: *Proc. 6th IRCOBI Conference on the Biomechanics of Impacts*, Salon de Provence, France, 166-175 (1981).

180. D. Otte *et al.*, Typical injuries to the soft body parts and fractures of the motorized 2-wheelers, in: *Proc. 6th IRCOBI Conference on the Biomechanics of Impacts*, Salon de Provence, France, 148-165 (1981).

181. H. Vallee *et al.*, Characteristics of objects struck by the head of moped rider or motorcyclists, in: *Proc. 6th IRCOBI Conference on the Biomechanics of Impacts*, Salon de Provence, France, 176-183 (1981).

182. B.P. Chinn *et al.*, *COST 327 Motorcycle Safety Helmets: A Literature Review*, Transport Research Laboratory, Croethrone TRL, UK, (1997).

183. J. Whitaker, *A Survey of Motorcycle Accidents*, TRRL Laboratory Report LR913, Crowthrone TRL, UK, (1980).

184. P.M. Fuller and J.N. Snider, Injury mechanisms in motorcycle accidents, in: *Proc. Int. IRCOBI Conf. Biomechanics of Impacts*, September, Birmingham, UK, 33-42 (1987).

185. D. Otte, A review of different kinematic forms in two-wheel-accidents – their influence on effectiveness of protective measures, in: *24th Stapp Car Crash Conf.*, Detroit, US, 561-605 (1980).

186. P.V. Hight *et al.*, An international review of motorcycle crashworthiness, in: *Proc. Int. IRCOBI Conference on Biomechanics of Impacts*, Zurich, 261-276 (1986).

187. K. Langweider, Collision characteristics and injuries to motorcyclists and moped drivers, in: *Proc. 21st Stapp Car Crash Conference*, New Orleans, 259-301 (1977).

188. A. Sporner *et al.*, *Development Of A Safety Concept For Motorcycles: Results Of Acccident Analysis And Of Crash Tests*, Jim Clarke Foundation, Edinbugh, UK, (1987).

189. D. Otte *et al.*, Erhebungen am unfallort; undall- und sicherheitsforschung strassenverkehr, *Heft*, **37**, 7-10 (1982).

190. D. Otte *et al.*, Typical injuries to the soft body parts and fractures of the motorized two-wheelers, in: *Proc. 6th IRCOBI Conf. On Biomechanics of Impacts*, Salon de Provence, France, 148-165 (1981).

191. P.L. Harms, Leg injuries and mechanisms in motorcycling accidents, in: *Proc. 12th Int. Conf. On Experimental Safety Vehicles*, Gothenburg, May, (1989).

192. A. Sporner, Risk of leg injuries of motorcyclists – present situation and countermeasures, in: *Proc. 12th Int. Conf. On Experimental Safety Vehicles*, Gotenberg, May, (1989).

193. B. Thomson and H. Rathgeber, Automated systems used for rapid and flexible generation of vehicle simulation models exemplified by a verified passenger car and a motorcycle model, in: *Proc. 8th IAVSD Vol. 12*, 645-54 (1983).

194. J. Happian-Smith, M.A. Macaulay and B.P. Chinn, Motorcycle Impact Simulation and Practical Verification, in: *Proc. 11th Experimental Safety Vehicles Conf.*, 858-865 (1987).

195. B.P. Chinn, J. Happian-Smith and M.A. Macaulay, The effect of leg protecting fairings on the overall motion of a motorcycle in a glancing impact, *Int. J. Impact Engng*, **8**(3), 265-279 (1989).

196. A.L. Yettram, J. Happian-Smith, L.S.M. Mo, M.A. Macaulay and B.P. Chinn, Computer simulation of motorcycle crash tests, *TRL Research Paper* P9407047, Presented at 14th Int. Conf. Enhanced Vehicle Safety, 23-26 May, Munich, (1994).

197. P.L.. Harms, *Crash Injury Investigation And Injury Mechnisms In Road Traffic Accidents*, TRL State of the Art Review, London: HMSO, (1993).

198. J.D. State, H.A. Jr Fenner, E.E. Flamboe, W.D. Nelson, L.N. Hames, Field application and research development of the abbreviated injury scale, in: *Proc. 15th Stapp Car Crash Conf.*, Nov 17-19, Coronado, California, 710-738 (1971).

199. D. Otte and G. Felten, Requirements on chin protection infull-face helmets for motorcyclist impact and injury situations, in: *Proc. 1991 Int. Motorcycle Conf.*, Bochum, Germany, 229-264 (1991).

200. T.A. Gennarelli, The state of the art of head injury biomechanics – a review, in: *Proc. 29th Conf. Association for Advancement of Automotive Medicine,* 447-463 (1985).

201. R. Lindenberg and E. Freytag, The mechanism of cerebral contusions, *Archives of pathology & Laboratory Medicine*, **69**, 440-469 (1960).

202. E.S. Gurdjian, Mechanisms of head injury, *Clinical Neurosurgery*, **12**, 112-128 (1966).

203. J.H. Adams *et al.*, Brain damage in non-missile head injury: observations in man and subhuman primates, in: *Recent Advances in Neuropathologym* W.T. Smith and J.B. Cacanagh, eds., Edinburgh, Churchill Livingstone, 165-190 (1982).

204. W. Trotter, Certain minor injuries of the brain, *Lancet*, **1**, 935-939 (1924).

205. D. Denny-Brown, Cerebral concusion, *Physiology Review,* **25**, 296-325 (1945).

206. A.K. Ommaya, The role of whiplash in cerebral concussion, in: *Proc. 10th Stapp Car Crash Conf.*, 314-324 (1969).

207. M.C. Lee, J.W. Melvin, *et al.*, Finite element analysis of traumatic subdural hematoma, in: *Proc. 31st Stapp Car Crash Conf.*, 67-77 (1987).

208. J.H. McElhaney, R.L. Stanlnaker and V.L. Roberts, Biomechanical aspects of head injury, in: *Proc. Symposium Human Impact Response*, W.F. King, H.J. Mertz, eds, Oct 2-3, General Motors Research Laboratories, Warren, Michigan, 85-112 (1972).

209. J.H. McElhaney, V.L. Roberts and J.W. Melvin, Biomechanics of seat belt design, in: *Proc. 16th Stapp Car Crash Conf.*, Nov 8-10, 321-344 (1972).

210. G.M. Mackay, P.F. Glyons, H.R.M. Hayes, D.K. Griffiths, and S.J. Rattenburg, Serious trauma to car occupants wearing seat belts, in: *Proc. 2nd Int. Conf. Biomechanics of Serious Trauma*, (1975).

211. H.R. Lissner and E.S. Gurdijan, *Experimental cerebral concussion*, ASME Report 60-WA-273, (1960).

212. E.S. Gurdjian, *et al.*, Quantitative determination of acceleration and intracranial pressure in experimental head injury, *Neurology*, **3**, 417-423 (1953).

213. E.S. Gurdjian, *et al.*, Intracranial pressure and acceleration accompanying head impacts in human cadavers, Surgery, Gynecology and Obstetrics, **113**, 185 (1961).

214. C.W. Gadd, Use of a weighted-impulse criterion for estimating injury hazard, in *Proc. 10th Stapp Car Crash Conf.*, 164-174 (1966).

215. A. Eiband, *Human tolerance to rapidly applied accelerations: a summary of the literature*, NASA Memorandum, Report No. 5-19-59E, June, (1959).

216. A. Slattenschek and W. Tauffkirchen, Critical evaluation of assessment methods for head impact applied in appraisal of brain injury hazard, in particular in head impact on windshields, in: *Proc. Int. Automobile Safety Conf. Compendium*, 1084-1112 (1970).

217. C.W. Gadd, Tolerable severity index in whole-head non-mechanical impact, in: *Proc. 15th Stapp Car Crash Conf.*, 164-174 (1971).

218. J. Versace, A review of the severity index, in: *Proc. 15th Stapp Car Crash Conf.*, 771-796 (1971).

219. V.R. Hodgson and L.M. Thomas, Effect of long duration impact on head, in: *Proc. 16th Stapp Car Crash Conf.*, 292-295 (1972).

220. G. Ryan *et al.*, Head impacts and brain injury in fatally injured pedestrains, in: *Proc. Int. IRCOBI Conf. On the Biokinetics of.Impacts*, 27-38 (1989).

221. F.J. Unterharnscheidt, Translational versus rotational acceleration – animal experiments with measured input, in: *Proc. 15th Stapp Car Crash Conf.*, Nov 17-19, Coronado, California, 767-782 (1971).

222. W. Johnson, A.G. Mamalis and S.R. Reid, Aspect of car design and human injury, in: *Human Body Dynamics: Impact, Occupational, and Athletic Aspects*, D.N. Ghista, ed., Oxford Medical Engineering Series, Clarendon Press, Oxford, 164-180 (1982).

223. I. Nassi and B. Schneiderman, Flowchart Techniques for Structured Programming, *ACM SIGPLAN Notices*, **8**(8), 12-26 (1973).

224. M.S.J. Hashmi, Ph.D. Thesis, University of Manchester, (1972).

225. N. Jones, Structural aspects of ship collisions, in *Structural Crashworthiness*, ed. N. Jones and T. Wierzbicki, Butterworths, London, 308-337 (1983).

226. W. Johnson and P.B. Mellor, *Engineering Plasticity*, Ellis Horwood Ltd., (1983).

227. Jr. W.D. Callister, *Materials Science and Engineering: An Introduction, 3rd Edition*, John Willey & Son, Inc., (1994).

228. D.C. Montgomery, *Design and Analysis of Experiments, 2nd Edition*, John Willey & Sons, Inc., (1984).

# Appendix A: FDM Components

## A.1 Syntax Definition

Before describing viewing individual section member's syntax, the way to describe a syntax needs to be clarified. Typical tag syntax definition will look like the following:

**Tag** type1 *Parameter1*, type2 *Parameter2* ;

Where

| | |
|---|---|
| **Tag** | The statement representation tag |
| type1 | The parameter type |
| *Parameter1* | The parameter value |
| , | Parameter delimiter |
| ; | Statement terminator |

The following table shows the parameter type and its definition.

Table A-1: Type definitions

| Type | Description |
|---|---|
| int | Integer value |
| ld | Long double or double |
| str | String |

## A.2 FDM PRE-PROCESSOR INPUT DATA FILE

Coding A-1 shows an example of a typical input data file.

```
WSV

SEGBEG
epc 1,1;
rem placing a remark
seg 0,                     0,                0,    0,    0, 0,    60, 0, 0;
seg shank&foot,            0,          0.47025, 0.03, 0.07, 6, 0.116, 0.03,
0.04;
seg thigh,    -0.2858479162947, 0.7560979162947, 0.03, 0.07, 5, 0.206, 0.03,
0.045;
seg abd&plv, -0.1458407736191, 0.8961050589689, 0.03, 0.07, 3, 0.281, 0.03,
0.09;
seg thorax,  0.05016922612591,  1.092115058713, 0.03, 0.07, 3, 0.216, 0.04,
0.08;
seg neck,     0.1108389879508,  1.152784820539, 0.03, 0.07, 2, 0.011, 0.03,
0.03;
seg head,     0.2625133925162,  1.304459225104, 0.03, 0.07, 2,  0.07, 0.02,
0.065;
prm knee,   6, 8, 160,12e8, 165,12e7, 170,80e2, 175,0, 330,0, 335,-80e2,
340,-50e7, 345,-50e8;
prm hip,   11, 8, 40,80e8, 45,80e7, 50,80e3, 60,0, 190,0, 200,-40e3, 205,-
50e7, 210,-50e8;
prm L4-S1, 12, 8, 165,15e8, 168,12e7, 170,60e3, 173,0, 182,0, 184,-60e3,
185,-18e7, 188,-28e8;
prm L3-L4, 13, 8, 165,15e8, 168,12e7, 170,60e3, 173,0, 182,0, 184,-60e3,
185,-18e7, 188,-28e8;
prm L1-L3, 14, 8, 165,15e8, 168,12e7, 170,60e3, 173,0, 182,0, 184,-60e3,
185,-18e7, 188,-28e8;
prm C4-T1, 18, 8, 155,50e8, 158,15e7, 160,20e4, 163,0, 215,0, 218,-10e4,
220,-25e7, 223,-80e8;
prm C1-C4, 19, 8, 155,50e8, 158,15e7, 160,20e4, 163,0, 215,0, 218,-10e4,
220,-25e7, 223,-80e8;
SEGEND

OBSBEG
pnt point1, 0.05, 0.5, 0.02;
OBSEND

DISBEG
box -0.50, -0.10, 1.70, 1.40;
typ 1;
DISEND

EXEBEG
dtm 0.000004;
ncy 25000;
inv 40, 0;
itc 1000;
EXEEND
```

Coding A-1: Typical FDM pre-processor input data file

The data file consists of four sections, which marked with respective delimiters, namely *Segment* section, *Obstacle* section, *Display* section, and *Execution* section. Each section consists of its own section members. And each section member is identified with a tag. Each section member carries important information. The section delimiters are printed in **bold**, section member tags are printed in *italic*, and section member parameters are printed in normal in the above example coding. The sequence of placing each segment is important, but not the sequence of section members inside a section. The section sequence has to follow from *Segment*, *Obstacle*, *Display*, to *Execution*. The sequence of placing the section member parameters is important. Parameters are delimited with " , " as the parameter delimiter. The definition of a section member has to be kept in a single line with " ; " as the terminator. Anything besides the defined keywords (the section delimiter and the section member tags) is classified as remarks, including the remark tag, *REM*.

## A.2.1 DESCRIPTION OF FILE SECTIONS

All members of the FDM pre-processor input data file are listed in this section.

A specific header is used for the FDM to recognise the file type. Furthermore, special instruction(s) can be placed along with the header. The header of the input data file starts with "WSV" and follows by any remarks about the data file.

### a. *Segment* Section

The *SEGBGN* marks the beginning and the *SEGEND* marks the end of the *Segment* section. This section contains four section members. They are used to describe the segment type, endpoint boundary conditions, segment definition, and pin joint resistive moment. The following table shows a summary of the *Segment s*ection tags definition.

Table A-2: Definition of Segment section tags

| Tag | Occurance* | Description |
|---|---|---|
| SEGBEG | Once | Marks the begin of the *Segment* section, a delimiter |
| typ | Multi, Last | Overall segment type, a *Segment* section member |
| seg | Multi | Segment definition, a *Segment* section member |
| prm | Multi | Pin joint definitions, including resistive moment, a *Segment* section member |
| epc | Multi, Last | End points' conditions, a *Segment* section member |
| SEGEND | Once | *Segment* section terminator, a delimiter |

## a.I Segment Section Declarer

**SEGBEG**

*Remarks:*

The declarer has to be assigned before defining any of its members. The section must end with the appropriate terminator

## a.II Segment Section Terminator

**SEGEND**

*Remarks:*

The terminator ends the *Segment* section begins with the declarer. Once the *SEGEND* is being place, any *Segment Tag* encountered after will be considered as remark.

## a.III Overall Segment Type Definition

**typ** int *x*;

* Once – restricted to appear only once. Last –only the latest definition is considered. Multi –Multiple definitions are allowed.

*Parameter Descriptions:*

$x$ – describes the segment type during the meshing and can have one of the following value:

| | |
|---|---|
| • 0 | no half link scenario |
| • 1 | half link at the beginning position, first link of the first segment only |
| • 2 | half link at the end position, last link of the last segment only |
| • 3 | half link at both beginning and end positions |

*Remarks:*

Can be defined several time in the *Segment* section, but only the latest one will be considered.

## a.IV Segment Definition

**SEG** str *desc*, ld $x$, ld $y$, ld $B$, ld $H$, int *node*, ld *mass*, ld $a$, ld, $b$;

*Parameter Descriptions:*

*desc* – place the description of the segment statement

$x$ – the x-coordinate of the end point of the segment

$y$ – the y-coordinate of the end point of the segment

$B$ – the width of the segment cross section

$H$ – the height of the segment cross section

*node* – number of nodes in the segment

*mass* – mass ratio over the total mass

$a$ – ellipsoid shape parameter for display purpose *b* – ellipsoid shape parameter for display purpose

*Remarks:*

The first *seg* definition is used to define the initial position / the begin of the overall structure. The total mass of the overall structure is defined in the first definition as well. For the first *seg* definition, the syntax will look like below:

SEG 0, ld *initX*, ld *initY*, , , , ld *overAllMass*, , ;

The above blank parameters are useless in describing the model. They can be left blank or simple having any useless value. *InitX*, *initY*, and *OverAllMass* are the initial co-ordinates of the initial position and the overall mass of the whole structure. The arrangement of the overall mass and mass ratio definition was designed to help describing the mass distribution of a human body.

## a.V Pin Joint Definition

**prm** str *desc*, int *node*, int *iSet*, ld *angle(1)*, ld *rMoment(1)*, ld *angle(2)*, ld *rMoment(2)*, ... ld *angle(iSet)*, ld *rMoment(iSet)*;

*Parameter descriptions:*

*desc* – description of the pin joint statement

*node* – the corresponding node number of the pin joint definition

*iSet* – The size of the angle-resistive moment data set, which is the number of angle-resistive moment data to define the angle-resistive moment relationship.

*angle(n)* – the *n* angle in degree with respect to the *n* resistive moment in the definition of angle-resistive moment

*rMoment(n)* – the *n* resistive moment with respect to the *n* angle in the definition of angle-resistive moment

*Remarks:*

When a pin joint is defined at a node, the links connected to the node will be freed to rotate. A set of angle-resistive moment information can be used to describe the resistive moment behavior of the node.

The angle here is referring to the rotary displacement from link *i* to link *i*+1 in counter clockwise direction, where the node connects link *i* with link *i*+1. The angle-resistive moment couples must be sorted in ascending order according to the angle.

### a.VI End Point Conditions

**epc** int *bgn*, int *end*;

*Parameter descriptions:*

*bgn* – the end point of the first segment

*end* – the end point of the last segment

The end point condition can have one of the following values:

- 1 free end, having all degrees of freedom
- 2 fixed end, restricting all degrees of freedom
- 3 pin joint, restricting the translational degrees of freedom, but free in rotation

## b. Obstacle Section

Only one type of section member is available. It is used to handle the contact between the structure with fixed point (global location). The FDM can handle multiple point obstacles at the same time.

### b.I Obstacle Section Declarer

**OBSBEG**

*Remarks:*

This declarer has the same characteristics with the other, which marks the beginning of *Obstacle* section and need a proper terminator.

### b.II Obstacle Section Terminator

**OBSEND**

### b.III Point Obstacle Definition

**pnt** str *desc*, ld *x*, ld *y*, ld *radius*;

*Parameter descriptions:*

*desc* – description of the point obstacle definition

*x* – the x-coordinate of the obstacle location in the global coordinate system

*y* – the y-coordinate of the obstacle location in the global coordinate system

*radius* – the effective radius of the contact zone with the point as its center

*Remarks:*

The contact will be handle by a contact algorithm designed by the author. The contact response will be taken into account when any of the link penetrates into the effective contact zone.

## c. Display Section

The *Display* section contains information about the screen display information. This section consists of two members, namely *Display Box* and *Display Type*.

### c.I Display Section Declarer

**DISBGN**

### c.II Display Section Terminator

**DISEND**

### c.III Display Box Definition

**box** ld *x1*, ld *y1*, ld *x2*, ld *y2*;

*Parameter Descriptions:*

*x1, y1* – coordinate of the lower point of the box

*x2, y2* – coordinate of the upper point of the box

*Remarks:*

The box defined is the box where the screen is going to be display in the pre-processor and the post-processor as well. The coordinate of the points is based on the global coordinate system.

### c.IV Display Type

**typ** int *dispType*;

*Parameter Descriptions:*

*dispType* – display type where

- 1 absolute displacement display
- 2 relative displacement display to the average human motion

## d. Execution Section

Simulation parameters are set in this section. All members of this section must be set in order for the FDM processor to carry its operations.

### d.I Execution Section Declarer

**EXEBEG**

### d.II Execution Section Terminator

**EXEEND**

### d.III Time Step Size

**dtm** ld *dt*;

*Parameter Descriptions:*

*dt* – the time step size in second

### d.IV Interested Time Step Interval

**itc** int *itc*;

*Parameter Descriptions:*

*itc* – number of time steps in an interval

*Remarks:*

Not all-very short time steps, *dt*, are of interest to the user. A collection of the predicted transient response will be stored in the processor output data file with large time frame, compare to the time step size. The *itc* is the interval size of each recorded result, in term of number of time steps.

### d.V Maximum Number of Time Step

**ncy** int *maxNTS*;

*Parameter Descriptions:*

*maxNTS* – maximum number of time step where the processor will terminate the processing iteration.

*Remarks:*

The *maxNTS* should be much greater than the interested time step interval, *itc*.

### d.VI Overall Velocity Definition

**inv** ld *velX*, ld *velY*;

*Parameter Descriptions:*

*velX* – velocity in x-axis

*velY* – velocity in y-axis

*Remarks:*

The *inv* definition provide the whole structure or body defined by the *segment* section a uniform velocity as the initial loading. Once defined, it applies to the whole body.

## A.3 FDM PROCESSOR OUTPUT DATA FILE

The binary data file is arranged as shown in

Figure A-1. The file header which consisting the data file independent overheads and then followed by those dependent overheads. Finally, the transient response data, which consists of all displacement, velocity, axial and shear forces, and bending moment history. The brackets "( ..)" used in the figure contain the data type carried. The "int" stands for integer and "ld" stands for long double.

## A.4 FDM MENU RESOURCES CODES

### A.4.1 FDM MAINFRAME MENU

Coding A-2 shows the resources coding of the FDM MainFrame menu system

### A.4.2 FDM PRE-PROCESSOR MENU

Coding A-3 shows the menu resources coding of the FDM Pre-processor.

### A.4.3 FDM POST-PROCESSOR MENU

Coding A-4 shows the menu resources coding of the FDM Post-processor.

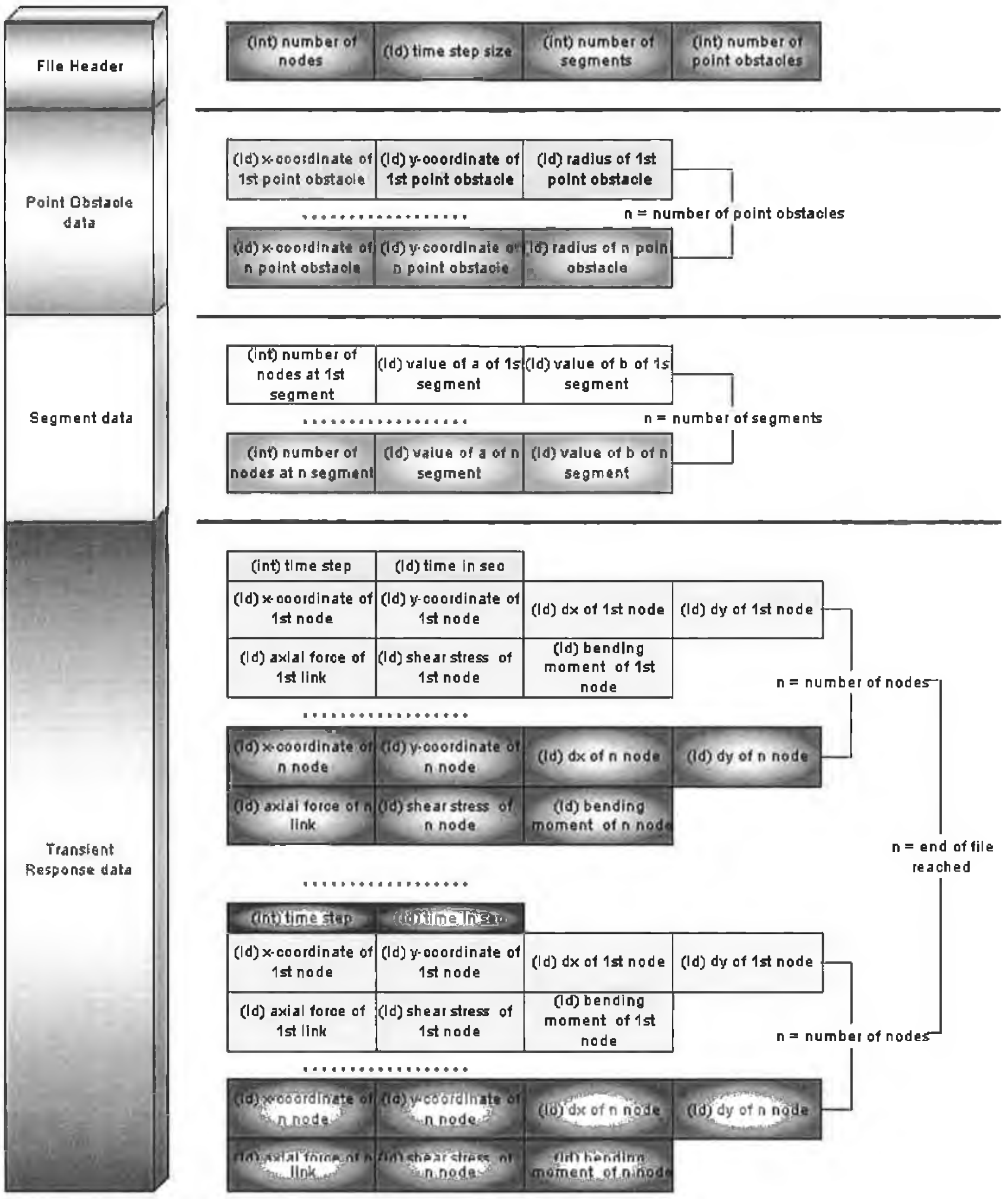

File Header
(int) number of nodes
(ld) time step size
(int) number of segments
(int) number of point obstacles
Point Obstacle data
(ld) x-coordinate of 1st point obstacle
(ld) y-coordinate of 1st point obstacle
(ld) radius of 1st point obstacle
n = number of point obstacles
(ld) x-coordinate of n point obstacle
(ld) y-coordinate of n point obstacle
(ld) radius of n point obstacle
Segment data
(int) number of nodes at 1st segment
(ld) value of a of 1st segment
(ld) value of b of 1st segment
n = number of segments
(int) number of nodes at n segment
(ld) value of a of n segment
(ld) value of b of n segment
Transient Response data
(int) time step
(ld) time in sec
(ld) x-coordinate of 1st node
(ld) y-coordinate of 1st node
(ld) dx of 1st node
(ld) dy of 1st node
(ld) axial force of 1st link
(ld) shear stress of 1st node
(ld) bending moment of 1st node
n = number of nodes
(ld) x-coordinate of n node
(ld) y-coordinate of n node
(ld) dx of n node
(ld) dy of n node
(ld) axial force of n link
(ld) shear stress of n node
(ld) bending moment of n node
n = end of file reached

Figure A-1: Post-processor binary data file system

```
IDR_MAINFRAME MENU PRELOAD DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&New\tCtrl+N",                ID_FILE_NEW
        MENUITEM "&Open Input File..\tCtrl+O",  ID_FILE_OPEN
        MENUITEM SEPARATOR
        MENUITEM "P&rint Setup...",             ID_FILE_PRINT_SETUP
        MENUITEM SEPARATOR
        MENUITEM "Recent File",                 ID_FILE_MRU_FILE1, GRAYED
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                       ID_APP_EXIT
    END
    POPUP "&View"
    BEGIN
        MENUITEM "&Toolbar",                    ID_VIEW_TOOLBAR
        MENUITEM "&Status Bar",                 ID_VIEW_STATUS_BAR
    END
    POPUP "&Help"
    BEGIN
        MENUITEM "&Help Topics",                ID_HELP_FINDER
        MENUITEM SEPARATOR
        MENUITEM "&About FDM...",               ID_APP_ABOUT
    END
END
```

Coding A-2: MainFrame menu code in FDM.rc

```
IDR_INPTYPE MENU PRELOAD DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&New\tCtrl+N",                 ID_FILE_NEW
        MENUITEM "&Open Input File...\tCtrl+O",  ID_FILE_OPEN
        MENUITEM "&Close",                       ID_FILE_CLOSE
        MENUITEM "&Save\tCtrl+S",                ID_FILE_SAVE
        MENUITEM "Save &As...",                  ID_FILE_SAVE_AS
        MENUITEM SEPARATOR
        MENUITEM "&Print...\tCtrl+P",            ID_FILE_PRINT
        MENUITEM "Print Pre&view",               ID_FILE_PRINT_PREVIEW
        MENUITEM "P&rint Setup...",              ID_FILE_PRINT_SETUP
        MENUITEM SEPARATOR
        MENUITEM "Recent File",                  ID_FILE_MRU_FILE1, GRAYED
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                        ID_APP_EXIT
    END
    POPUP "&Edit"
    BEGIN
        MENUITEM "&Undo\tCtrl+Z",                ID_EDIT_UNDO
        MENUITEM SEPARATOR
        MENUITEM "Cu&t\tCtrl+X",                 ID_EDIT_CUT
        MENUITEM "&Copy\tCtrl+C",                ID_EDIT_COPY
        MENUITEM "&Paste\tCtrl+V",               ID_EDIT_PASTE
    END
    POPUP "Run FDM"
    BEGIN
        MENUITEM "Node && Initial Cond.",        ID_RUN_NODEINIT
        MENUITEM "FDM",                          ID_FDM
    END
    POPUP "&View"
    BEGIN
        MENUITEM "&Toolbar",                    ID_VIEW_TOOLBAR
        MENUITEM "&Status Bar",                 ID_VIEW_STATUS_BAR
    END
    POPUP "&Window"
    BEGIN
        MENUITEM "&New Window",                 ID_WINDOW_NEW
        MENUITEM "&Cascade",                    ID_WINDOW_CASCADE
        MENUITEM "&Tile",                       ID_WINDOW_TILE_HORZ
        MENUITEM "&Arrange Icons",              ID_WINDOW_ARRANGE
    END
    POPUP "&Help"
    BEGIN
        MENUITEM "&Help Topics",                ID_HELP_FINDER
        MENUITEM SEPARATOR
        MENUITEM "&About FDM...",               ID_APP_ABOUT
    END
END
```

Coding A-3: Pre-Processor menu code in FDM.rc

```
IDR_OPTTYPE MENU PRELOAD DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&New\tCtrl+N",                ID_FILE_NEW
        MENUITEM "&Open Input File...\tCtrl+O", ID_FILE_OPEN
        MENUITEM "&Close",                      ID_FILE_CLOSE
        MENUITEM "&Save\tCtrl+S",               ID_FILE_SAVE
        MENUITEM "Save &As...",                 ID_FILE_SAVE_AS
        MENUITEM SEPARATOR
        MENUITEM "&Print...\tCtrl+P",           ID_FILE_PRINT
        MENUITEM "Print Pre&view",              ID_FILE_PRINT_PREVIEW
        MENUITEM "P&rint Setup...",             ID_FILE_PRINT_SETUP
        MENUITEM SEPARATOR
        MENUITEM "Recent File",                 ID_FILE_MRU_FILE1, GRAYED
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                       ID_APP_EXIT
    END
    POPUP "&Edit"
    BEGIN
        MENUITEM "&Undo\tCtrl+Z",               ID_EDIT_UNDO
        MENUITEM SEPARATOR
        MENUITEM "Cu&t\tCtrl+X",                ID_EDIT_CUT
        MENUITEM "&Copy\tCtrl+C",               ID_EDIT_COPY
        MENUITEM "&Paste\tCtrl+V",              ID_EDIT_PASTE
    END
    POPUP "Output Result"
    BEGIN
        MENUITEM "Displacement Path",           ID_OTP_DISPLACEMENT
        MENUITEM "Animate Path",                ID_OTP_ANIMATEPATH
    END
    POPUP "&View"
    BEGIN
        MENUITEM "&Toolbar",                    ID_VIEW_TOOLBAR
        MENUITEM "&Status Bar",                 ID_VIEW_STATUS_BAR
    END
    POPUP "&Window"
    BEGIN
        MENUITEM "&New Window",                 ID_WINDOW_NEW
        MENUITEM "&Cascade",                    ID_WINDOW_CASCADE
        MENUITEM "&Tile",                       ID_WINDOW_TILE_HORZ
        MENUITEM "&Arrange Icons",              ID_WINDOW_ARRANGE
    END
    POPUP "&Help"
    BEGIN
        MENUITEM "&Help Topics",                ID_HELP_FINDER
        MENUITEM SEPARATOR
        MENUITEM "&About FDM...",               ID_APP_ABOUT
    END
END
```

Coding A-4: FDM post-processor menu resource coding in FDM.rc

## A.5 EXAMPLE OF A FDM PROCESSOR TEXT-BASED OUTPUT DATA FILE

```
node, x, y, N, Q, M

Cycle: 1000,  time: 0.4000e-2
1, 0.1897e0, -0.2858e-2, 0.0000e0, 0.0000e0, 0.0000e0
2, 0.1669e0, 0.7955e-1, 0.4427e5, -0.1154e6, -0.9863e4
3, 0.1440e0, 0.1619e0, 0.6551e5, -0.6471e5, -0.1540e5
4, 0.1207e0, 0.2442e0, 0.9543e5, 0.7069e5, -0.9352e4
5, 0.9727e-1, 0.3265e0, 0.9541e5, 0.4495e5, -0.5509e4
6, 0.7361e-1, 0.4089e0, 0.1032e6, 0.6423e5, 0.0000e0
7, 0.3357e-1, 0.4791e0, 0.1039e6, 0.1739e5, 0.1406e4
8, -0.6214e-2, 0.5495e0, 0.8980e5, -0.1113e6, -0.7597e4
9, -0.4615e-1, 0.6198e0, 0.6625e5, 0.6621e4, -0.7062e4
10, -0.8632e-1, 0.6900e0, 0.4047e5, -0.1794e5, -0.8512e4
11, -0.1267e0, 0.7601e0, 0.1350e5, 0.1051e6, 0.0000e0
12, -0.7977e-1, 0.8065e0, -0.8718e5, 0.0000e0, 0.0000e0
13, -0.3207e-1, 0.8522e0, -0.7387e5, 0.0000e0, 0.0000e0
14, 0.1526e-1, 0.8982e0, -0.4174e5, 0.0000e0, 0.0000e0
15, 0.8094e-1, 0.9632e0, -0.6095e5, 0.6566e5, 0.6067e4
16, 0.1467e0, 0.1028e1, -0.1780e5, -0.4706e4, 0.5632e4
17, 0.2126e0, 0.1093e1, -0.7502e5, -0.5663e5, 0.3998e3
18, 0.2432e0, 0.1123e1, 0.1336e6, -0.9321e4, 0.0000e0
19, 0.2723e0, 0.1154e1, -0.1879e6, 0.0000e0, 0.0000e0
20, 0.3330e0, 0.1215e1, -0.1441e5, -0.2042e4, -0.1752e3
21, 0.3937e0, 0.1276e1, 0.6585e4, 0.2042e4, -0.3553e-14

Cycle: 2000,  time: 0.8000e-2
1, 0.3208e0, -0.8240e-3, 0.0000e0, 0.0000e0, 0.0000e0
2, 0.2619e0, 0.6117e-1, 0.2039e5, 0.1442e6, 0.1233e5
3, 0.2044e0, 0.1244e0, 0.3946e5, 0.1326e6, 0.2366e5
4, 0.1483e0, 0.1889e0, 0.9494e5, 0.4770e4, 0.2406e5
5, 0.9517e-1, 0.2560e0, 0.1141e6, -0.1177e6, 0.1399e5
6, 0.6038e-1, 0.3355e0, 0.1815e6, -0.1615e6, 0.0000e0
7, 0.3272e-1, 0.4115e0, 0.2533e6, 0.1011e5, 0.8166e3
8, 0.2972e-1, 0.4922e0, 0.4814e6, 0.5581e5, 0.5322e4
9, 0.3249e-1, 0.5731e0, 0.5166e6, 0.8628e5, 0.1230e5
10, 0.3743e-1, 0.6538e0, 0.4156e6, 0.5728e5, 0.1694e5
11, 0.4383e-1, 0.7349e0, 0.3524e6, -0.2082e6, 0.0000e0
12, 0.8286e-1, 0.7884e0, 0.2812e6, 0.0000e0, 0.0000e0
```

```
13, 0.1207e0, 0.8427e0, 0.2856e6, 0.0000e0, 0.0000e0
14, 0.1635e0, 0.8929e0, 0.7940e5, 0.0000e0, 0.0000e0
15, 0.2308e0, 0.9562e0, 0.1547e6, 0.4819e4, 0.4453e3
16, 0.2981e0, 0.1020e1, 0.2099e6, 0.1082e6, 0.1045e5
17, 0.3656e0, 0.1083e1, 0.1383e6, -0.1081e6, 0.4554e3
18, 0.3969e0, 0.1112e1, 0.7133e6, -0.1061e5, 0.0000e0
19, 0.4304e0, 0.1138e1, -0.1028e7, 0.0000e0, 0.0000e0
20, 0.4868e0, 0.1203e1, 0.1299e6, -0.7346e4, -0.6304e3
21, 0.5433e0, 0.1268e1, -0.1448e6, 0.7348e4, 0.0000e0

Cycle: 3000,  time: 0.1200e-1
1, 0.2494e0, 0.5201e-2, 0.0000e0, 0.0000e0, 0.0000e0
.......................................................
.......................................................
```

Coding A-5: Typical FDM processor text-based output data file

# Appendix B: MSS Components

## B.1 General Description of the MSS Pre-Processor Classes

The following table shows all the classes incorporated into the MSS Pre-processor.

Table B-1: MSS pre-processor classes description

| Class | Type | Description |
|---|---|---|
| CAboutDlg | Interface | Handles of About dialog box |
| CArc | Object | Holds an arc's attributes |
| CChildFrame | Application | MDI controls |
| CCircle | Object | Holds a circle's attributes |
| CContact | Object | Holds a contact element 's attributes |
| CDlgCoinNd | Interface | Handles of coincide node elimination dialog box |
| CDlgdtCal | Interface | Handles of critical time step size dialog box |
| CDlgGenMData | Interface | Handles of a general multi couple data entering dialog box |
| CDlgLkPivotCrt | Interface | Handles of link pivot creation dialog box |
| CDlgLnCt | Interface | Handles of line construction dialog box |
| CDlgMatAsgn | Interface | Handles of material assignment dialog box |
| CDlgNdPt | Interface | Handles of node construction from point dialog box |
| CDlgPointCt | Interface | Handles of point construction dialog box |
| CDlgPtNd | Interface | Handles of point construction from node dialog box |
| CDlgResMmt | Interface | Handles of revolute joint data entry dialog box |
| CEndLk | Object | Holds a end link condition |
| CLink | Object | Holds a link's attributes |
| CLkPivot | Object | Holds a link pivot attributes |
| CLn | Object | Holds a line attributes |
| CLnCnt | Object | Holds a line constraint attributes |
| CMainFrame | Application | Window main frame control |
| CMatrix | Base | Matrix formulation and implementations |
| CMSSPreApp | Application | Initialization class |
| CMSSPreDoc | Application | Document class |
| CMSSPreView | Application | View class |
| CNdCons | Object | Holds a node use specified conditions |
| CNode | Object | Holds a node attributes |
| CPPgArcGeo | Interface | Handles of arc geometry definition property page |
| CPPgCirGeo | Interface | Handles of circle geometry definition property page |
| CPPgContNdLk | Interface | Handles of contact element node – link specification property page |
| CPPgContType | Interface | Handles of contact type and friction model selection property page |

| | | |
|---|---|---|
| CPPgFDSv | Interface | Handles of processor information entry property page |
| CPPgListSel | Interface | Handles of selection list display property page |
| CPPgLkEnd | Interface | Handles of end link boundary condition selection property page |
| CPPGLKGeo | Interface | Handles of link geometry data display property page |
| CPPgLnCnt | Interface | Handles of line constraints entry property page |
| CPPgLNFD | Interface | Handles of line cross section dimension, meshing and material definition property page |
| CPPgLnGeo | Interface | Handles of line geometry definition property page |
| CPPgMatGen | Interface | Handles of material model definition property page |
| CPPgMultiNdLkSel | Interface | Handles of multiple nodes and links selection property page |
| CPPgNDDOF | Interface | Handles of nodal constraints definition property page |
| CPPgNDGeo | Interface | Handles of nodal geometry display property page |
| CPPgNdInit | Interface | Handles of initial nodal loading definition property page |
| CPPgNdLk | Interface | Handles of node-link selection property page |
| CPPgPSContConst | Interface | Handles of contact constant definition property page |
| CPPgPSContConst2 | Interface | Handles of contact constant definition property page |
| CPPgSelect | Interface | Handles of hierarchy tree selection property page |
| CPPgTapper | Interface | Handles of line tappering definition property page |
| CPPgWSVar | Interface | Handles of workspace variables entry property page |
| CPreMaterial | Object | Holds a material model attributes |
| CPShContact | Interface | Handles of contact element definition property sheet |
| CPShDel | Interface | Handles of general deletion property sheet |
| CPShGen | Interface | Handles of general property sheet |
| CPShLN | Interface | Handles of line definition property sheet |
| CPt | Object | Holds a point attributes |
| CResMmt | Object | Holds a revolute joint attributes |
| CVector | Base | Vector formulation and implementations |

## B.2 MSS PRE-PROCESSOR OBJECT TYPE CLASSES

All member variables of the object type classes will be present under this section.

### a. CArc Class

The Carc class is used to handle arc shape in geometrical modelling. An arc is defined by a centre point, radius, starting rotary position and ending rotary position.

*Attributes:*

*center* – unsigned integer. Center point reference number, points to the corresponding point.

*start* – double. Starting angle of the arc referring to the horizontal of the global x-y coordinate system. Direction of the rotation reference is set with variable *direction*. In degree.

*end* – double. Ending angle of the arc referring to the horizontal of the global x-y coordinate system. Direction of the rotation reference is set with variable *direction*. In degree.

*radius* – double. Radius of the arc from its center point.

*iNode* – unsigned integer. Number of nodes in the whole arc.

*B* – double. Width of the cross section dimension.

*H* – double. Height of the cross section dimension.

*iMat* – unsigned integer. Pointer to the corresponding material number.

*direction* – integer. Indication of both the *start* and the *end* rotary direction.

- 0 counter clock wise
- 1 clock wise

## b. CCircle Class

The CCircle class is used to describe circles in the geometrical construction of the model. The circle shape can represent a hollow tube or a solid bar. Option is available for these representations. A circle is represented by two entities, the centre point and the radius.

*Attributes:*

*center* – unsigned integer. Center point reference number, points to the corresponding point.

*radius* – double. Radius of the arc from its center point.

*hollow* – Boolean. Options for hollow tube or solid bar

- true hollow tube
- false solid bar

*iNode* – unsigned integer. Number of nodes in the whole arc.

*B* – double. Width of the cross section dimension.

*H* – double. Height of the cross section dimension.

*iMat* – unsigned integer. Pointer to the corresponding material number.

*RiNode* – unsigned integer. Radial number of nodes.

## c. CContact Class

The CContact class holds information about a contact element. This class is designed to hold general contact-impact element. A contact element consists of three nodes (for the case of 2D contact), which forms a contact surface and a contact node. The contact and friction models have to be specified during the construction of the contact element. Respective contact model coefficients and friction model coefficients are stored in this class as well.

*Attributes:*

*iNd* – unsigned integer. Pointer to the contact node.

*iLk* – unsigned integer. Pointer to the contact surface, represented by a link element.

*Kn* – double. Contact stiffness, used by Penalty Method of contact model.

*iLkI* – unsigned integer. Link pointer of which connected to node I of the contact element.

*iLkJ* – unsigned interger. Link pointer of which connected to node J of the contact element.

*ContactType* – contact type. Contact model. The contact type can be one of the following:

| | |
|---|---|
| • CT_PENALTY | Penalty Method |
| • CT_LAGRANGE | Lagrange Multiplier |
| • CT_THICKNESSPENALTY | Penalty Method with thickness |
| • CT_THICKNESSLAGRANGE | Lagrange Multiplier with thickness |
| • CT_KINEMATICS | Kinematics Contact Method |
| • CT_ADJPENALTY | Adaptive Penalty Method |
| • CT_KIN_ADJPENALTY | Combined Kinematics-Adaptive Penalty Method |

*alpha* – double. Multiplier factor, <1. Used in CT_LAGRANGE and CT_THICKNESSLAGRANGE.

*epsil* – double. Penetration where the multiplier takes no effect. A User-defined compatibility tolerance. Used in CT_LAGRANGE and CT_THICKNESSLAGRANGE.

*thickness* – double. The effective thickness of the contact element. No effect for CT_PENALTY and CT_LAGRANGE.

*frictionType* – friction type. Friction model. The friction type can have one of the following values:

| | |
|---|---|
| • FT_FRICTIONLESS | frictionless mode |
| • FT_ELASTICCOULOMB | non-classical friction model where microslip is allowed |
| • FT_RIGIDCOULOMB | classical Coulomb's friction model |

*miu* – double. Coefficient of friction in classical Coulomb's friction model. Valid with FT_RIGIDCOULOMB.

*Kt* – double. Sticking stiffness. Valid with FT_ELASTICCOULOMB.

*fact* – double. Dynamic over static friction factor, normally < 1.

*restCoef* – double. Coefficient of restitution. Must be < 1. Valid with CT_KINEMATICS.

## d. CEndLk Class

The CEndLk class stores information about boundary conditions at the open edge of the structure, where an opened-end link is.

*Attributes:*

*pLink* – pointer of CLink class. Pointer of the end link.

*nodeNo* – integer. Open end node number of the end link. The value refers to the node of the Clink class pointed by the *pLink*. Possible values are 1 and 2.

*type* – end link type. The end link type can be one of the following:

| | |
|---|---|
| • EL_FIXED | fixed end |
| • EL_PIVOTED | pivoted end |
| • EL_FREED | freed end |

## e. CLink Class

The CLink class is used to store attributes of a link element. A link consists of two nodes.

*Attributes:*

*ipt1* – unsigned integer. Pointer to the first node of the link definition.

*ipt2* – unsigned integer. Pointer to the second node of the link definition.

*iNode* – unsigned integer. Number of nodes in the whole line.

*iMat* – unsigned integer. Pointer to the corresponding material number.

*Desc* – CString. Description of the link

*B* – double. Width of the cross section dimension.

*H* – double. Height of the cross section dimension.

## f. CLkPivot Class

A pivot joint is a simpler version of a revolute joint, where no rotary limits and no resistive moment.

*Attributes:*

*NdType* – byte. First or second node of the link definition is pivoted. Possible values are 1 and 2.

*iLk* – unsigned integer. Pointer to the corresponding link.

## g. CLn Class

The CLn class holds the properties of a line in the geometry model. A line connects two points. To cope with tapering configuration along a line, a pair of cross section dimension is used. The cross section dimension of the first point is stored in the first set of dimension and wise versus. Linear interpolation is used for the cross section dimension in between the two points.

*Attributes:*

*ipt1* – unsigned integer. Pointer to the first point of the line definition.

*ipt2* – unsigned integer. Pointer to the second point of the line definition.

*iNode* – unsigned integer. Number of nodes in the whole line.

*iMat* – unsigned integer. Pointer to the corresponding material number.

*Desc* – CString. Description of the line

*BBegin* – double. Width of the cross section dimension at point 1.

*BEnd* – double. Width of the cross section dimension at point 2.

*HBegin* – double. Height of the cross section dimension at point 1.

*HEnd* – double. Height of the cross section dimension at point 2.

## h. CLnCnt Class

Constraints can be set on a line, where the line length and its orientation in rotation are defined. The line construction constraint is not as intelligence as commercial CAD system, but serve the purpose of the author's requirement. The entities (the length and orientation) of the line will be updated when an update line constraints message is sent. A fixed point is required where the location of the point will be fixed as the reference.

*Attributes:*

*iLn* – unsigned integer. Pointer to the corresponding line.

*fPoint* – integer. Fixed point of the line. Can be either 1 or 2.

*Length* – double. Length of the line

*theta* – double. Angle between the line and the horizon of global Cartesian coordinate system (horizontal on the right-hand side of the center point). Counter clockwise is defined as positive. In radian from -π to π.

## i. CNdCons Class

The CNdCons class holds any special nodal constraints in degrees of freedom. The degrees of freedom are three in linear motions and respective bending rotary motions. Thus, a total of six degrees of freedom can be constrained by this class.

*Attributes:*

*iNode* – unsigned integer. Pointer to the corresponding node.

*Ux, Uy, Uz* – Boolean. Constraints of linear degrees of freedom in x, y, and z axes of the global Cartesian coordinate system.

*Mx, My, Mz* – Boolean. Release of bending moment in x, y, and z axes of the global Cartesian coordinate system.

## j. CNode Class

The CNode class holds attributes of a node. The attributes are the nodal displacement and initial loading, which includes the initial velocity and constant force. The CNode class inherits from the CPt class, which describes a point attributes. The location of the node is stored in the inherited attributes of the CPt class, which is publicly inherited from the CVector class. CNode.i, CNode.j, and CNode.k will store the global location of the node with respect to the global Cartesian coordinate system.

*Atrributes:*

*vel* – CVector. Initial velocity of the node.

*force* – CVector. Constant force loading of the node.

### k. CPreMaterial Class

The pre-processor material model is defined and stored in CPreMaterial class. Material model can be multi-linear elastic – multi-linear elastic plastic. The stress-strain relation is defined by several pair of Young's modulus – upper stress relationships.

*Attributes:*

*Rho* – double. Density of the material.

*D* – double. coefficient of strain rate sensitivity model

*P* – double. coefficient of strain rate sensitivity model.

*iSubFlg* – unsigned integer. Number of sub-flanges.

*iFlg* – unsigned integer. Number of flanges.

*E[]* – array of double. Young's modulus.

*Sigma[]* – array of double. Upper stress value with respective Young's modulus.

### l. CPt Class

The CPt class stores the location of a point. The CPt class is a public inherit of CVector. No extra member variable is included.

### m. CResMmt Class

The CResMmt stores information about a revolute joint. Angle - resistive moment relationship is defined by a set of angles – resistive moment pair data.

*Attributes:*

*iLk1* – unsigned integer. Pointer to the first link of the revolute joint connected links.

*ILk2* – unsigned integer. Pointer to the second link of the revolute joint connected links.

*iSet* – integer. Number of degree – resistive moment pairs.

*degree[]* – array of double. Degree from link *iLk1* to link *iLk2* in counter clockwise. Valid value from 0 to 360.

*moment[]* – array of double. Resistive moment at respective degree of rotation in *degree*.

*nUpper* – double. Negative (clockwise) upper limit of the revolute rotary limit. In degree.

*pLower* – double. Positive (counter clockwise) upper limit of the revolute rotary limit. In degree.

*dampCoef* – double. Damping coefficient.

## B.3 MENU DEFINITIONS OF THE MSS PRE-PROCESSOR

### a. MainFrame Menu System

The MainFrame menu system is shown as the window menu bar at the initial stage of the MSS pre-processor execution, where no specific document is opened. Which means, no child frame is available for the user to work on. The user will have to open a new document or an existing one from the hard storage (hard disk or micro disk). The menu provides the most basic functions, which open document and exit the application play the main roles. The following coding shows the menu organisation and respective command message identity (ID).

```
IDR_MAINFRAME MENU PRELOAD DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&New\tCtrl+N",                ID_FILE_NEW
        MENUITEM "&Open...\tCtrl+O",            ID_FILE_OPEN
        MENUITEM SEPARATOR
        MENUITEM "P&rint Setup...",             ID_FILE_PRINT_SETUP
        MENUITEM SEPARATOR
        MENUITEM "Recent File",                 ID_FILE_MRU_FILE1, GRAYED
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                       ID_APP_EXIT
    END
    POPUP "&View"
    BEGIN
        MENUITEM "&Toolbar",                    ID_VIEW_TOOLBAR
        MENUITEM "&Status Bar",                 ID_VIEW_STATUS_BAR
    END
    POPUP "&Help"
    BEGIN
        MENUITEM "&Help Topics",                ID_HELP_FINDER
        MENUITEM SEPARATOR
        MENUITEM "&About MSSPre...",            ID_APP_ABOUT
    END
END
```

Coding B-1: MSS pre-processor MainFrame menu coding from MSSPre.rc

## b. Execution Menu

The Execution menu system is activated after the document is available to be used, for generating and/or editing models. The menu ID is *IDR_MSSPRETYPE*. The menu system consists of all the functions required from building the model to generating the MSS processor data file. It contains all the functions shown in Figure 4-2. The following coding shows the menu organization and respective command message identity (ID).

```
IDR_MSSPRETYPE MENU PRELOAD DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&New\tCtrl+N",                ID_FILE_NEW
        MENUITEM "&Open...\tCtrl+O",            ID_FILE_OPEN
        MENUITEM "&Close",                      ID_FILE_CLOSE
        MENUITEM "&Save\tCtrl+S",               ID_FILE_SAVE
        MENUITEM "Save &As...",                 ID_FILE_SAVE_AS
        MENUITEM SEPARATOR
        MENUITEM "Import",                      ID_FILE_IMPORT
        MENUITEM "Insert",                      ID_FILE_INSERT
        MENUITEM "Export",                      ID_FILE_EXPORT
        MENUITEM SEPARATOR
        MENUITEM "&Print...\tCtrl+P",           ID_FILE_PRINT
        MENUITEM "Print Pre&view",              ID_FILE_PRINT_PREVIEW
        MENUITEM "P&rint Setup...",             ID_FILE_PRINT_SETUP
        MENUITEM SEPARATOR
        MENUITEM "Recent File",                 ID_FILE_MRU_FILE1, GRAYED
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                       ID_APP_EXIT
    END
    POPUP "&Edit"
    BEGIN
        MENUITEM "&Undo\tCtrl+Z",               ID_EDIT_UNDO
        MENUITEM SEPARATOR
        MENUITEM "Cu&t\tCtrl+X",                ID_EDIT_CUT
        MENUITEM "&Copy\tCtrl+C",               ID_EDIT_COPY
        MENUITEM "&Paste\tCtrl+V",              ID_EDIT_PASTE
    END
    POPUP "&Geometry"
    BEGIN
        POPUP "&Construct"
        BEGIN
            MENUITEM "&Point",               ID_GEOMETRY_CONSTRUCT_POINT
            MENUITEM "Point fm Nd",          ID_GEOMETRY_CONSTRUCT_POINTFMND
            MENUITEM "&Line",                ID_GEOMETRY_CONSTRUCT_LINE
            MENUITEM "&Arc",                 ID_GEOMETRY_CONSTRUCT_ARC
            MENUITEM "Circle",               ID_GEOMETRY_CONSTRUCT_CIRCLE
            MENUITEM SEPARATOR
        END
        POPUP "Edit"
        BEGIN
            MENUITEM "Point",                   ID_GEOMETRY_EDIT_POINT
            MENUITEM "Line",                    ID_GEOMETRY_EDIT_LINE
            MENUITEM "Arc",                     ID_GEOMETRY_EDIT_ARC
            MENUITEM "Circle",                  ID_GEOMETRY_EDIT_CIRCLE
        END
        POPUP "Delete"
        BEGIN
            MENUITEM "Point",                   ID_GEOMETRY_DELETE_POINT
            MENUITEM "Line",                    ID_GEOMETRY_DELETE_LINE
            MENUITEM "Arc",                     ID_GEOMETRY_DELETE_ARC
            MENUITEM "Circle",                  ID_GEOMETRY_DELETE_CIRCLE
        END
        MENUITEM SEPARATOR
        POPUP "Line Constraints"
        BEGIN
            MENUITEM "Create",            ID_GEOMETRY_LINECONSTRAINTS_CREATE
            MENUITEM "Edit",              ID_GEOMETRY_LINECONSTRAINTS_EDIT
            MENUITEM "Delete",            ID_GEOMETRY_LINECONSTRAINTS_DELETE
        END
        MENUITEM "Apply Constraints",           ID_GEOMETRY_APPLYCONSTRAINTS
        MENUITEM SEPARATOR
        MENUITEM "WorkSpace Var",               ID_GEOMETRY_WORKSPACEVAR
    END
    POPUP "Display"
    BEGIN
        MENUITEM "Point No",                    ID_DISPLAY_POINTNO
        MENUITEM "Line No",                     ID_DISPLAY_LINENO
        MENUITEM SEPARATOR
        MENUITEM "List Points",                 ID_DISPLAY_LISTPOINTS
        MENUITEM "List Lines",                  ID_DISPLAY_LISTLINES
        MENUITEM SEPARATOR
        MENUITEM "Nodes",                       ID_DISPLAY_NODES
        MENUITEM "Links",                       ID_DISPLAY_LINKS
        MENUITEM "Node No",                     ID_DISPLAY_NODENO
        MENUITEM "Link No",                     ID_DISPLAY_LINKNO
    END
    POPUP "Contact"
    BEGIN
        POPUP "Point-Surface"
        BEGIN
            MENUITEM "Create",          ID_CONTACT_POINTSURFACE_CREATE
            MENUITEM "Sel Nodes",       ID_CONTACT_POINTSURFACE_SELNODES
            MENUITEM "Sel Links",       ID_CONTACT_POINTSURFACE_SELLINKS
            MENUITEM "MultiSelect",     ID_CONTACT_POINTSURFACE_MULTISELECT
            MENUITEM SEPARATOR
            MENUITEM "Edit",            ID_CONTACT_POINTSURFACE_EDIT
            MENUITEM "Delete",          ID_CONTACT_POINTSURFACE_DELETE
            MENUITEM "Delete All",      ID_CONTACT_POINTSURFACE_DELETEALL
        END
        MENUITEM SEPARATOR
        POPUP "Joint Constraint"
        BEGIN
            MENUITEM "Create",          ID_CONTACT_JOINTCONSTRAINT_CREATE
            MENUITEM "Edit",            ID_CONTACT_JOINTCONSTRAINT_EDIT
            MENUITEM "Delete",          ID_CONTACT_JOINTCONSTRAINT_DELETE
        END
        POPUP "Link Pivot"
        BEGIN
            MENUITEM "Create",          ID_CONTACT_LINKPIVOT_CREATE
            MENUITEM "Delete",          ID_CONTACT_LINKPIVOT_DELETE
        END
    END
    POPUP "FD Generation"
    BEGIN
        MENUITEM "NodeLink",                    ID_GENERATE_NODELINK
        MENUITEM SEPARATOR
        POPUP "Node"
        BEGIN
```

```
                MENUITEM "Create",                  ID_FDGENERATION_NODE_CREATE
                MENUITEM "Delete",                  ID_FDGENERATION_NODE_DELETE
                MENUITEM "Update",                  ID_FDGENERATION_NODE_UPDATE
            END
            MENUITEM "Link",                        ID_FDGENERATION_LINK
            MENUITEM SEPARATOR
            MENUITEM "Coincide Nodes (auto)",      ID_FDGENERATION_COINCIDENODES
            MENUITEM "Nodes Redundancy (slc)",   ID_FDGENERATION_NODESREDUNDANCY
            MENUITEM "Coincide Node (dlg)",      ID_FDGENERATION_COINCIDENODEDLG
            POPUP "Redundancies"
            BEGIN
                MENUITEM "Node Cond",      ID_FDGENERATION_REDUNDANCIES_NODECOND
                MENUITEM "EndLink",        ID_FDGENERATION_REDUNDANCIES_ENDLINK
            END
            POPUP "Conditions"
            BEGIN
                MENUITEM "Node",                 ID_FDGENERATION_CONDS_NODE
                MENUITEM "Link",                 ID_FDGENERATION_CONDS_LINK
            END
            MENUITEM SEPARATOR
            POPUP "Material Prop"
            BEGIN
                MENUITEM "Create",          ID_FDGENERATION_MATERIALPROP_CREATE
                MENUITEM "Copy",            ID_FDGENERATION_MATERIALPROP_COPY
                MENUITEM "Edit",            ID_FDGENERATION_MATERIALPROP_EDIT
                MENUITEM "Delete",          ID_FDGENERATION_MATERIALPROP_DELETE
            END
            MENUITEM "Assign Material",        ID_FDGENERATION_ASSIGNMATERIAL
            MENUITEM SEPARATOR
            MENUITEM "Critical Time Step",     ID_FDGENERATION_CRITICALTIMESTEP
            MENUITEM "MSS Processor File",         ID_FDGENERATION_FDFILE
        END
        POPUP "&View"
        BEGIN
            MENUITEM "&Toolbar",                    ID_VIEW_TOOLBAR
            MENUITEM "&Status Bar",                 ID_VIEW_STATUS_BAR
        END
        POPUP "&Window"
        BEGIN
            MENUITEM "&New Window",                 ID_WINDOW_NEW
            MENUITEM "&Cascade",                    ID_WINDOW_CASCADE
            MENUITEM "&Tile",                       ID_WINDOW_TILE_HORZ
            MENUITEM "&Arrange Icons",              ID_WINDOW_ARRANGE
        END
        POPUP "&Help"
        BEGIN
            MENUITEM "&Help Topics",                ID_HELP_FINDER
            MENUITEM SEPARATOR
            MENUITEM "&About MSSPre...",            ID_APP_ABOUT
        END
    END
```

Coding B-2:MSS pre-processor IDR_MSSPRETYPE menu coding in MSSPre.rc

### c. Pop-up Menu

A pop-up menu is design to handle right mouse button click message. The menu will pop-up if a user click on right-hand side mouse button at the proper time. The user can then decide what to select from the list of the menu.

Coding B-3 shows the menu items and their respective command message ID.

```
IDR_MENU_RBUTTONUP MENU DISCARDABLE
BEGIN
    MENUITEM "Properties",                    ID_PROPERTIES, INACTIVE
    MENUITEM SEPARATOR
    MENUITEM "Point",                         ID_RBUTTONUP_POINT
    MENUITEM "Line",                          ID_RBUTTONUP_LINE
    MENUITEM "Node",                          ID_RBUTTONUP_NODE
    MENUITEM "Link",                          ID_RBUTTONUP_LINK
END
```

Coding B-3: MSS pre-processor IDR_MENU_RBUTTONUP menu coding in MSSPre.rc

## B.4 MSS PRE-PROCESSOR DATA FILE SYSTEM

The file system used to store activities of the MSS Pre-processor is documented in this section. Figure below shows the file structure.

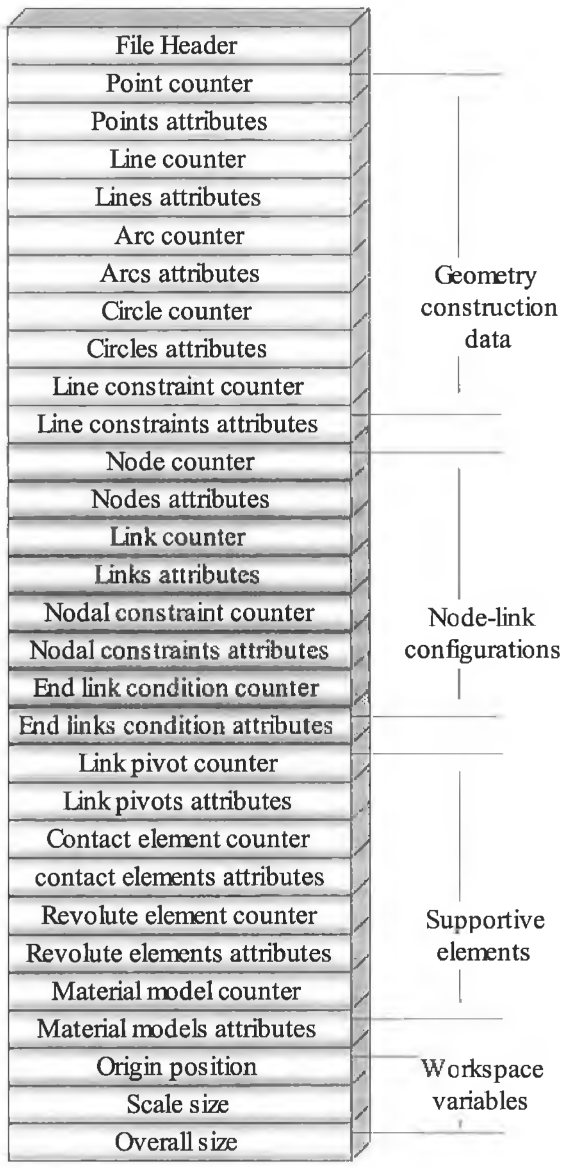


Figure B-1: MSS pre-processor Data File Structure

The data file header is stored in text form and reads "WSV Pre-processor data file". This is to help a user recognizing the data file type by opening the file by any kind of editor, may it be text or hexidecimal viewer. Geometry construction data will first to be stored. They are consisting of all attributes of points, lines, arcs, and circle. The line constraints are considered in the categories of geometry construction. Before storing any kind of entities, the respective counter will be placed. The entities are stored as a whole of respective call, where all the attributes of the respective class will be placed. For example, a *Line* (an entity) attributes consist of CLn class attributes with the same sequence shown in "CLn" under sub-section of B.2 of Appendix B. The expended view of *Line counter* and *Lines entities* is shown in the following figure.

| |
|---|
| (unsigned integer) Line Counter |
| (CLn) Line attributes [0] |
| (CLn) Line attributes [1] |

•••••••••••••

| |
|---|
| (CLn) Line attributes [Line Counter-2] |
| (CLn) Line attributes [Line Counter-1] |

Figure B-2: Part of MSS pre-processor data file for storing line entities

The brackets contain the data type of the information. The *Line Counter* is an unsigned integer, while the *Line* entities are group under the CLn class type. The square bracket contains the array number, which is the identity of a line. The rest is following the same pattern.

## B.5 ENTITY DEFINITIONS OF THE MSS PRE-PROCESSOR COMPONENT DATA FILE

All entities used in the MSS Pre-processor Component Data File are present in the following sub-sections. The syntax convention is similar as described in section A.1 of Appendix A.

### a. Point Entity

**Pt** int *PtNo*, double *x*, double *y*;

*Parameter Descriptions:*

*PtNo* – point number, which represents the sequence of the point stored in the memory. MSS pre-processor will ignore this attribute while reading.

*x* – the x-component of the point location.

*y* – the y-component of the point location.

## b. Line Entity

**Ln** int *LnNo*, int *iPt1*, int *ipt2*, double *BBegin*, double *HEnd*,
double *HBegin*, double *HEnd*, int *iNode*, int *iMat*, str *desc*;

*Parameter Descriptions:*

*LnNo* – line number, which representing the sequence of the line stored in the memory. MSS pre-processor will ignore this attribute while reading.

*iPt1* – pointer to a point as the first point describing the line.

*iPt2* – pointer to a point as the second point describing the line.

*BBegin* – width in cross section dimension at the first point describing the line.

*BEnd* – width in cross section dimension at the second point describing the line.

*HBegin* – height in cross section dimension at the first point describing the line.

*HEnd* – height in cross section dimension at the second point describing the line.

*iNode* – number of nodes along the line.

*iMat* – material model number.

*desc* – description of the line

*Remarks:*

The value of *iPt1 and iPt2* is refer the sequence of the point being created within the data file. If *ipt1* = 5, means referring to the sixth point being defined throughout the data file. The indexing sequence starts from zero. The same kind of pointing convention is used throughout the data file. Thus, *iMat* is defined through the same way.

## c. Arc Entity

**Arc** int *ArcNo*, int *iCenterPt*, double *radius*, double *startRadian*, double *endRadian*, int *direction*, double *B*, double *H*, int *iNode*, int *iMat*;

*Parameter Descriptions:*

*ArcNo* – arc number, which representing the sequence of the arc stored in the memory. MSS pre-processor will ignore this attribute while reading.

*iCenterPt* – pointer to a point which represent the center of the arc.

*radius* – the radius of the arc.

*startRadian* – starting angle from the horizon in radian.

*endRadian* – ending angle from the horizon in radian.

*Direction* – rotary direction of both the *startRadian* and *endRadian*. The value can be one of the following:

- 0 counter clockwise
- 1 clockwise

*B* – width in cross section dimension.

*H* – height in cross section dimension.

*iNode* – number of nodes along the arc.

***iMat*** – material model number.

## d. Circle Entity

**Cir** int *CirNo*, int *iCenterPt*, double *radius*, Boolean *hollow*, double *B*, double *H*, int *iNode*, int *RiNode*, int *iMat*;

*Parameter Descriptions:*

*CirNo* – circle number, which representing the sequence of the circle stored in the memory. MSS pre-processor will ignore this attribute while reading.

*iCenterPt* – pointer to a point which represent the center of the circle.

*radius* – the radius of the circle.

*hollow* – hollow or solid option. Having one of the following values:

| | |
|---|---|
| • t | hollow tube |
| • f | solid bar |

*B* – width in cross section dimension.

*H* – height in cross section dimension.

*iNode* – number of nodes along the circle.

*RiNode* – number of nodes in radial direction from the center of the circle.

*iMat* – material model number.

*Remarks:*

The *RiNode* parameter will be ignored if *hollow* = t.

## e. Line Constraint Entity

**LnCnt** int *LnCntNo*, int *iLine*, int *iFixedPt*, double *length*,
double *angleRadian*;

*Parameter Descriptions:*

*LnCntNo* – number of line constraint, which representing the sequence of the line constraint stored in the memory. MSS pre-processor will ignore this attribute while reading.

*iLine* – pointer to the corresponding line.

*iFixedPt* – the point of the line which its location is fixed. The possible values are:

- 1 the first point of the line
- 2 the second point of the line

*length* – length of the line.

*angleRadian* – the angle of the line from the horizon in radian and counter clock wise as positive rotation.

*Remarks:*

The other point, which is not fixed, will be changed according to the *length* and *angleRadian* referring to the fixed point

## f. Node Entity

**Node** int *NodeNo*, double *x*, double *y*, double *z*, double *Vx*,
double *Vy*, double *Vz*, double *Fx*, double *Fy*, double *Fz*;

*Parameter Descriptions:*

*NodeNo* – node number, which representing the sequence of the node stored in the memory. MSS pre-processor will ignore this attribute while reading.

*x, y, z* – location of the node in x-, y-, and z-axes, respectively.

*Vx, Vy, Vz* – initial velocity of the node in x-, y- and z-axes, respectively.

*Fx, Fy, Fz* – constant external force applied on the node in x-, y- and z-axes, respectively.

## g. Link Entity

**Link** int *LkNo*, int *iNd1*, int *iNd2* double *B*, double *H*, int *iMat*;

*Parameter Descriptions:*

*LkNo* – link number, which representing the sequence of the link stored in the memory. MSS pre-processor will ignore this attribute while reading.

*iNd1* – pointer to a node as the first node describing the line.

*iNd2* – pointer to a node as the second node describing the line.

*B* – width in cross section dimension.

*H* – height in cross section dimension.

*iMat* – material model number.

## h. Nodal Constraint Entity

**SPoint** int *SPNo*, Boolean *Ux*, Boolean *Uy*, Boolean *Uz*,
Boolean *Mx*, Boolean *My*, Boolean *Mz*;

*Parameter Descriptions:*

*SPNo* – Nodal constraint number, which representing the sequence of the nodal constraint stored in the memory. MSS pre-processor will ignore this attribute while reading.

*Ux, Uy, Uz* – linear motion constraint in global x-, y- and z-axes, respectively. Possible values are t for true and f for false. Default setting is false.

*Mx, My, Mz* – releasing of bending moment in global x-, y- and z-axes, respectively. Possible values are t for true and f for false. Default setting is false.

## i. End Link Condition Entity

**EndLk** int *ElkNo*, int *iLk*, int *iNd*, int *iType*;

*Parameter Descriptions:*

*ElkNo* – end link condition number, which representing the sequence of the end link condition stored in the memory. MSS pre-processor will ignore this attribute while reading.

*iLk* – pointer to the corresponding link.

*iNd* – indicates first or second node of the link. Possible values are 1 for first node and 2 for second node of the corresponding link.

*iType* – end link type at the specified node. Possible values are as the following:

- 1 fixed
- 2 supported
- 3 freed

## j. Link Pivot Joint Entity

**LkPivot** int *LkPivotNo*, int *iLk*, int *iNd*;

*Parameter Descriptions:*

*LkPivotNo* – link pivot joint number, which representing the sequence of the link pivot joint stored in the memory. MSS pre-processor will ignore this attribute while reading.

*iLk* – pointer to the corresponding link.

*iNd* – indicates first or second node of the link. Possible values are 1 for first node and 2 for second node of the corresponding link.

## k. Revolute Joint Entity

**RMmt** int *RMmtNo*, int *iLk1*, int *iLk2*, double *dampCoef*, int *iSet*, double *pLower*, double *nUpper*, double *degree(0)*, double *moment(0)*, double *degree(1)*, double *moment(1)*, ...

*Parameter Descriptions:*

*RMmttNo* – revolute joint number, which representing the sequence of the revolute joint stored in the memory. MSS pre-processor will ignore this attribute while reading.

*iLk1* – pointer to the first corresponding link.

*iLk2* – pointer to the second corresponding link.

*dampCoef* – damping coefficient, in moment-s/rad.

*iSet* – number of angle-resistive moment relation pairs to describe the characteristics of the resistive moment.

*pLower* – counter clockwise rotary limit, 0° - 360°.

*nUpper* – clockwise rotary limit, 0° - 360°.

***degree(i), moment(i)*** – *ith* pair of the degree-resistive moment relationships. The pair represents the resistive moment of *moment(i)* at degree of *degree(i)* for the revolute joint.

## l. Contact Element Entity

**Ctact** int ***CtactNo***, int ***iNode***, int ***iLink***, int ***contactType***,, int ***frcitionType***, double ***Kn***, double ***Kt***, double ***miu***, double ***fact***, double ***thickness***, double ***restCoef***;

*Parameter Descriptions:*

***CtactNo*** – contact element number, which representing the sequence of the contact element stored in the memory. MSS pre-processor will ignore this attribute while reading.

***iNode*** – pointer to the corresponding node as the contact node of the contact element.

***iLink*** – pointer to the corresponding link as the contact surface of the contact element.

***contactType*** – contact model, where

- 0 CT_PENALTY, Penalty Method
- 1 CT_LAGRANGE, Lagrange Multiplier Method
- 2 CT_THICKNESSPENALTY, Penalty Method with thickness
- 3 CT_THICKNESSLAGRANGE, Lagrange Multiplier Method with thickness
- 4 CT_KINEMATICS, Kinematics Contact-impact Method.
- 5 CT_ADJPENALTY, Adaptive Penalty Method.
- 6 CT_KIN_ADJPENALTY, Combined Kinematics – Adaptive Penalty Method

***frictionType*** – friction model, where

- 0 FT_FRICTIONLESS, frictionless.

- 1 FT_ELASTICCOULOMB, non-classical elastic Coulomb's friction model.
- 2 FT_RIGIDCOULOMB, classical Coulomb's friction model.

*Kn* – contact stiffness.

*Kt* – sticking stiffness

*miu* – coefficient of friction

*fact* – dynamic over static friction factor.

*thickness* – effective thickness.

*RestCoef* – coefficient of restitution.

## m. Material Entity

**Mat** int *MatNo*, double *Rho*, int *iFlg*, int *iSubFlg*, double *D*, double *P*, double *E(1)*, double *sigma(1)*, double *E(2)*, double *sigma(2)*, ... ;

*Parameter Descriptions:*

*MatNo* – material number, which representing the sequence of the material model stored in the memory. MSS pre-processor will ignore this attribute while reading.

*Rho* – density.

*iFlg* – number of flanges.

*iSubFlg* – number of sub-flanges.

*D* – coefficient of the strain rate sensitivity model.

*P* – coefficient of the strain rate sensitivity model.

*E[i], Sigma[i]* – *i*th pair of Young's modulus – upper stress relationships used to describe the stress-strain relationship in the material model. The maximum value of *i* depends on the number of sub-flanges, *iSubFlg*.

## B.6 EXAMPLE OF THE MSS PRE-PROCESSOR COMPONENT DATA FILE

Coding B-4 shows a cropped typical component data file.

```
WSV MSSPre exported component data text file
by MSS Pre-processor version 1.2, copyright reserved, Oct 1999
----------------------------------
Base MSS Pre-processor file: D:\WongSV\Ph.D. Project\VC project\MSS
Data\Human Bike\corr strength.pre
Generated on Monday, October 25, 1999  at 14:19:31

*** Points' position ***
- Format: Pt (int PtNo), (double x),(double y);
Pt 0, -0.5,0.02;
Pt 1, -0.7508,0.02;
Pt 2, -0.734145,0.0821573;
.........................................

*** Lines' properties ***
- Format: Ln (int LnNo), (int iPt1),(int iPt2),
   (double BBegin),(double BBegin), (double HBegin),(double HBegin),
   (int iNode), (int iMat), (string Desc);
Ln 0, 0,1, 0.0422178,0.0422178, 0.07,0.07, 4, 0,       ;
Ln 1, 1,2, 0.0626824,0.0626824, 0.07,0.07, 2, 1,       ;
.................................

*** Arcs' properties ***
- Format: Arc (int ArcNo), (int iCenterPt),(double radius),
   (double startRadian),(double endRadian),(int direction),
   (double B),(double H), (int iNode), (int iMat);

*** Circles' properties ***
- Format: Cir (int CirNo), (int iCenterPt),(double radius),
   (bool hollow), (double B),(double H), (int iNode),
   (int RiNode), (int iMat);

*** Lines' constraint ***
- Format: LnCnt (int LnCntNo), (int iLine),(int iFixedPt),
   (double length),(double angleRadian);
LnCnt 0, 0,1, 0.2508,3.14159;
LnCnt 1, 1,1, 0.06435,1.309;
LnCnt 2, 2,1, 0.4059,1.22173;
..........................................

*** Nodes defination ***
```

```
- Format: Node (int NodeNo), (double x),(double y),(double z)
    (double Vx),(double Vy),(double Vz),
    (double Fx),(double Fy),(double Fz);
Node 0, -0.5,0.02,0, 15,0,0, 0,0,0;
Node 1, -0.5836,0.02,0, 15,0,0, 0,0,0;
..............................................

*** Links defination ***
- Format: Link (int LkNo), (int iPt1),(int iPt2),
    (double B),(double H), (int iMat);
Link 0, 0,1, 0.03,0.07, 0;
Link 1, 1,2, 0.03,0.07, 0;
......................................

*** SPoints (node constraints) defination ***
- Format: SPoint (int SPNo), (bool Ux),(bool Uy),(bool Uz),
    (bool Mx),(bool My),(bool Mz);
SPoint 0, 4, f,f,f, t,t,t;
SPoint 1, 9, f,f,f, t,t,t;
.....................................

*** End Link conditions ***
- Format: EndLk (int ELkNo), (int iLk),(int iNode),(int iType);
- iType: 1=fixed, 2=support, 3=free
EndLk 0, 0, 1, 3;
EndLk 1, 23, 2, 3;
......................................

*** Link Pivot Joint ***
- Format: LkPivot (int iLk), (byte NdType),(int iNode);
- NdType: 1=1st node, 2=2nd node

*** Resistive Moments defination ***
- Format: RMmt (int RMmtNo), (int iLk1),(int iLk2), (double dampCoef),
    (int iSet),
    (double pLower),(double pUpper), (double nLower),(double nUpper),
    (double degree0),(double moment0),
    (double degree1),(double moment1), ..;
RMmt 0, 3,4, 0, 10, 290,360, 0,125, 0,500, 90,210, 110,60, 130,15, 170,0,
255,0, 265,-5, 285,-30, 305,-200, 360,-500;
..............................

*** Contact conditions ***
- Format: Ctact (int CtactNo), (int iNode),(int iLink),
    (int contactType), (int frictionType), (double Kn), (double Kt)
    (double miu), (double fact), (double thickness), (double restCoef);
Ctact 0, 35, 53, 6, 0, 5e+007, 0, 0.15, 0.9, 0.05, 0;
Ctact 1, 35, 54, 6, 0, 5e+007, 0, 0.15, 0.9, 0.05, 0;
.....................................

*** Material types ***
- Format: Mat (int MatNo), (double Rho),(int iFlg),(int iSubFlg),
    (double D),(double P), (double E1),(double Sigma1), ..;
Mat 0, 1700, 8,2, 1000, 0.65, 1.74e+009,1.35e+008, 8e+008,1.45e+008;
Mat 1, 1700, 8,2, 1000, 0.65, 1.74e+009,1.35e+008, 8e+008,1.45e+008;
.....................................

*** END PRX FILE***
```

Coding B-4: Cropped MSS pre-processor component data file example

## B.7 FUNCTION IMPLEMENTATIONS OF THE MSS PRE-PROCESSOR

Important functions of the MSS Pre-processor are explained and respective implementation is presented in the following sub-sections.

### a. Geometry Entities

Main geometry entities are consisting of *point*, *line*, *arc* and *circle*. Whereas l*ine* c*onstraint* is the geometry support entity. Three basic functions are required to maintain the entities. They are entity construction, edition and deletion.

#### a.I Construction

The construction function has a general form of implementation. Figure B-3 shows the message and data flow diagram while constructing a geometry entity. The number in the brackets indicates the event sequence. While a construction message is fired, the message will be captured by one or more of the class, which is called, *Message Catcher* class in the diagram. The message will route to a handling function, normally a member function of the *Message Catcher* class. An instance of appropriate interface type class will be created, and the corresponding interface (may it be windows, dialog box or others) will be displayed on the screen and awaiting for the user to respond. Data will be entered and messages about the change in the interface will be fired and captured by the *Interface* class. Appropriate response will be taken towards the interface message. Update of the interface will be the most common response. After termination of the interface, the data will be sent to the interface class. The data may undergo some sort of interpretation before being transferred in the handling function. The data will then be used to add new entity to the CMSSPreDoc data. The view or the display of the application window will be updated to show the new entity.

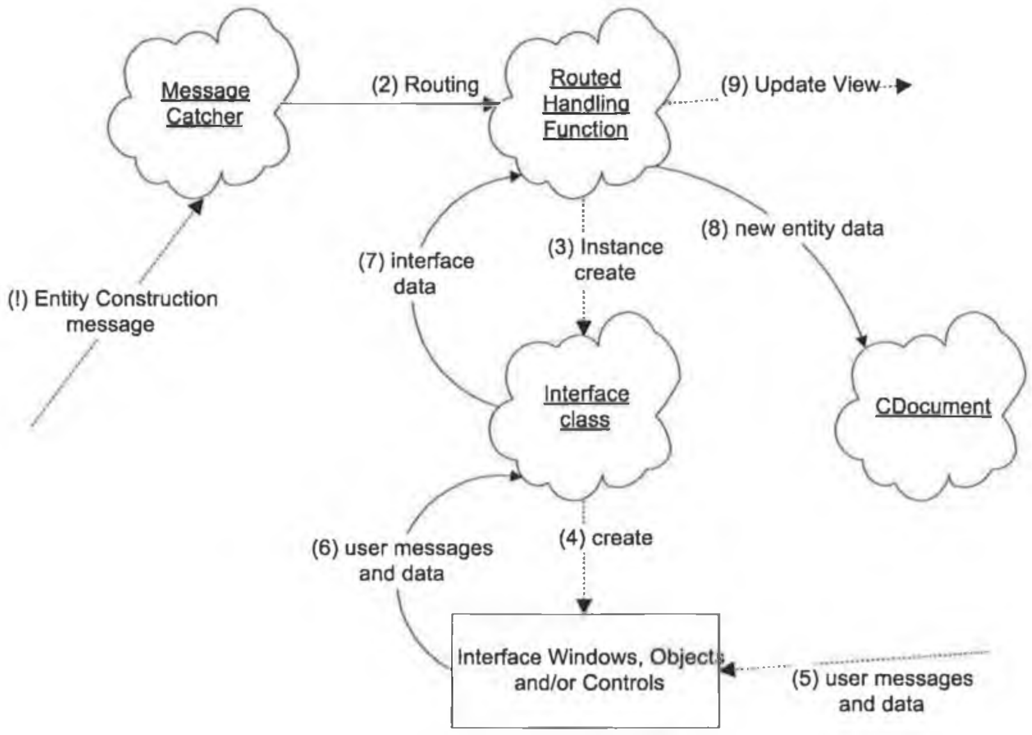
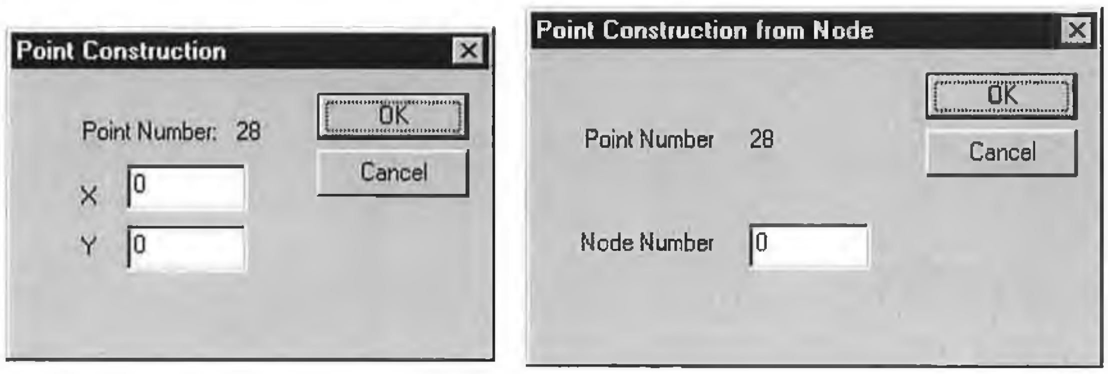


Figure B-3: Geometry construction message and data flow diagram

*Point*

A point's attribute is simply its location. A user has two ways to create a point in MSS pre-processor, construct from scratch which can be achieved by selecting *Geometry – Construct – Point* in the menu bar, or construct from a reference node by selecting *Geometry – Construct – Point from Nd*. The first action will fire a command message, namely *ID_GEOMETRY_CONSTRUCT_POINT*, and the latter will be *ID_GEOMETRY_CONSTRUCT_POINTFMND*. Both command messages are captured by the CMSSPreView class, and their respective routed handling functions are *CMSSPreView::OnGemetryConstructPoint* and *CMSSPreView::OnGeometryConstructPointfmnd*. Their respective interface control classes are CDlgPointCt and CDlgPtNd classes. The dialog box of point constructions are shown below:


Point Construction

Point Number: 28

X 0

Y 0

Point Construction from Node

Point Number 28

Node Number 0

Figure B-4: Point construction dialog boxes. Left: Scratch Construction. Right: Node based construction.

The first method requires a user to give co-ordinate location of the point. The second method creates a point from an existing node. The new point reference, point number, is displayed on both the dialog boxes. The implementation of both the *CMSSPreView::OnGemetryConstructPoint* and *CMSSPreView::OnGeometryConstructPointfmnd* is simple and can be generalised with the following Nassi-Schneiderman chart. In fact all the MSS pre-processor entity construction implementations are using the similar method. Minor changes are expected from one implementation to another.

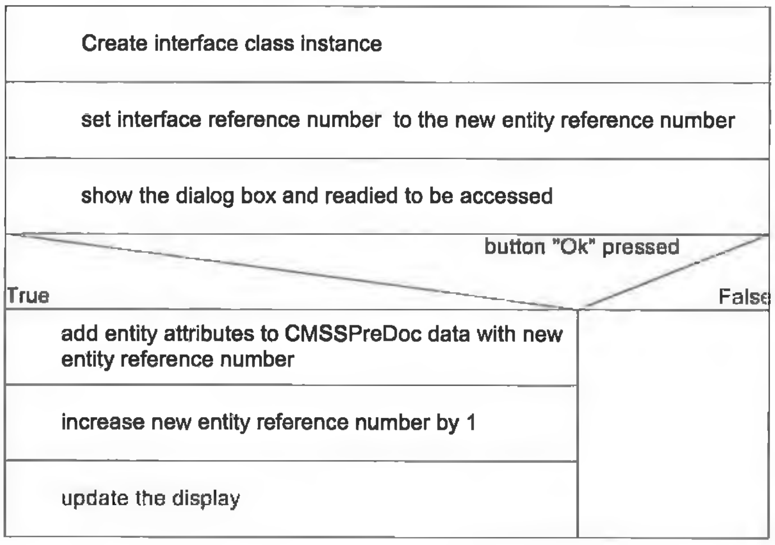

Create interface class instance
set interface reference number to the new entity reference number
show the dialog box and readied to be accessed
button "Ok" pressed
True
False
add entity attributes to CMSSPreDoc data with new entity reference number
increase new entity reference number by 1
update the display

Figure B-5: MSS pre-processor entity construction implementation chart

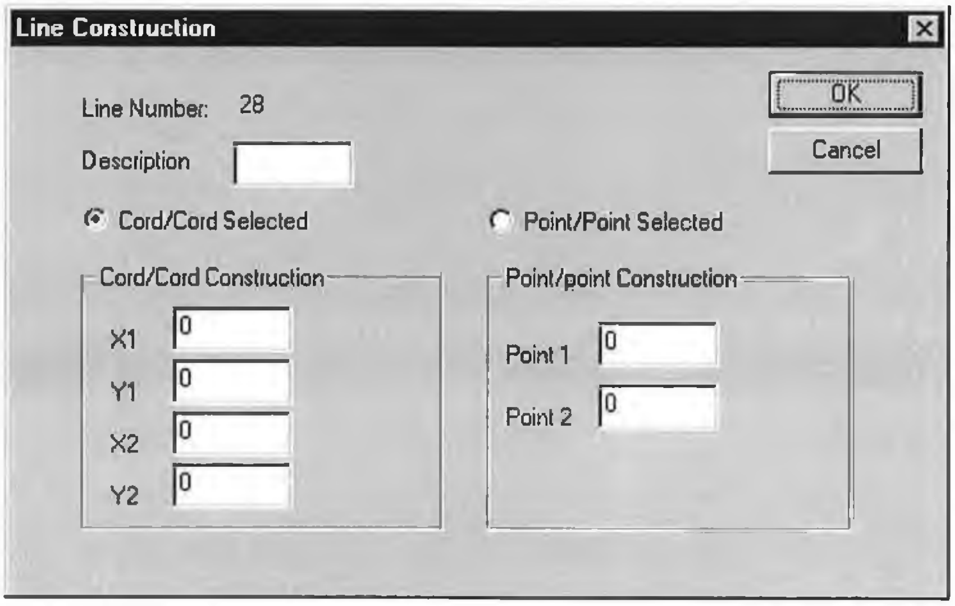
*Line*

A line consists of two points. Thus, to define a line, a user required to specify its member points. The command message ID to construct a line is *ID_GEOMETRY_CONSTRUCT_LINE*. The message is captured by MSSPreView class and handled by function MSSPreView::OnGeometryConstructLine. The interface class is CDlgLnCt which controls the dialog box shown in Figure B-6. A user can either create two new points by providing respective coordinate location, or select existing points by providing their point reference numbers.

*Arc*

An arc needs its center point and its radius. Besides that, starting point and ending point of the arc needs to be defined. The command message ID to construct an arc is *ID_GEOMETRY_CONSTRUCT_ARC*. The message is captured by MSSPreView class and handled by function MSSPreView::OnGeometryConstructArc. The interface handling is a bit different. Arc construction combines the geometry construction with modelling construction of the arc. In modelling construction, the user will have to provide number of nodes to be placed on the arc, cross section dimension and material model reference number. The interface is not an ordinary dialog box, but consisting of two property pages, under control of CPPgArcGeo class and CPPgLNFD class. The first class controls the geometry construction while the other for modelling information. These two classes are bind under CPShGen class, controlling the property sheet. Figure B-7 shows the display of the property page of arc geometry construction. The other will be discussed later.

A user can place a point reference number if an existing point is used to be the center point of the arc. He/she can also create a new point entity directly. When giving the starting angle and ending angle, please note the rotating directions, where CCW means counter clockwise and CW means clockwise. The angle starts from the horizontal line on the right hand side of the center point.


Line Construction

Line Number: 28

Description

[✓ Cord/Cord Selected] Point/Point Selected

Cord/Cord Construction

X1 0

Y1 0

X2 0

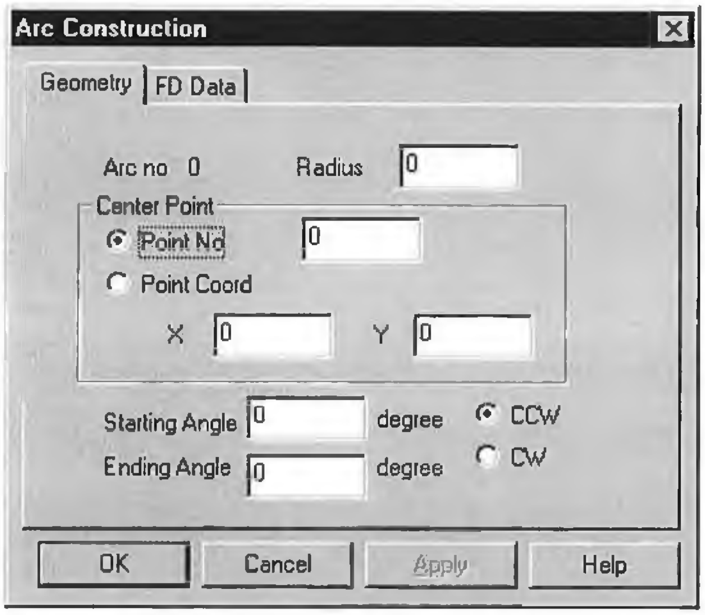
Y2 0

Point/point Construction

Point 1 0

Point 2 0

Figure B-6: Line construction dialog box


Arc Construction

Geometry | FD Data

Arc no 0

Radius 0

Center Point

[✓ Point No] 0

Point Coord

X 0 Y 0

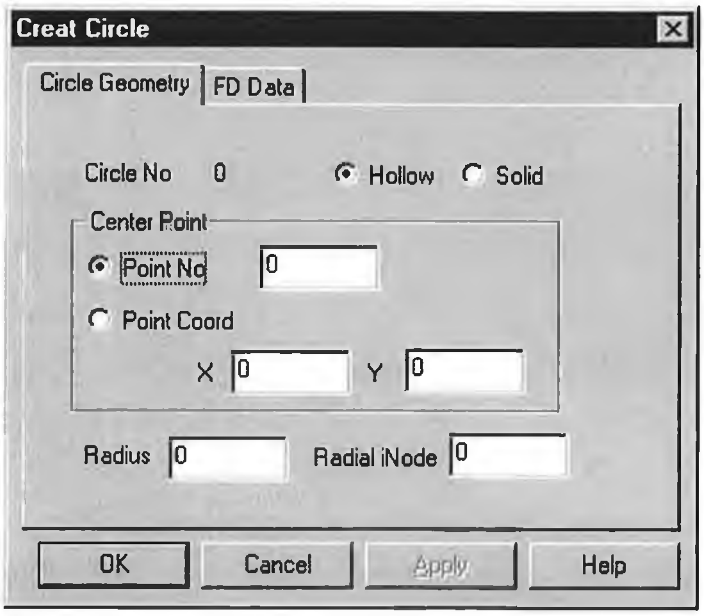
Starting Angle 0 degree [✓ CCW]

Ending Angle 0 degree CW

Figure B-7: Arc geometry construction property page

*Circle*

A circle simply requires a center point and radius. The command message ID to construct an arc is *ID_GEOMETRY_CONSTRUCT_CIRCLE.* The message is captured by MSSPreView class and handled by function MSSPreView::OnGeometryConstructCIRCLE. The interface handling is similar to arc construction interface, where a property sheet consisting two property pages is used. The interface classes involve are CPPgCirGeo class, CPPgLNFD class and CPShGen class. The first two are controlling geometry construction property page and modelling parameter property page respectively. The last class controls the interaction between these two pages and the whole property sheet as well. The property page of circle geometry construction is shown in Figure B-8. The other will be discussed later. Similar to arc geometry construction, a user can use existing point as the center point or create new point entity. A user can also select to construct either a hollow tube or a solid bar.


Creat Circle

Circle Geometry | FD Data

Circle No 0 [✓ Hollow] Solid

Center Point
[✓ Point No] 0
Point Coord
X 0 Y 0

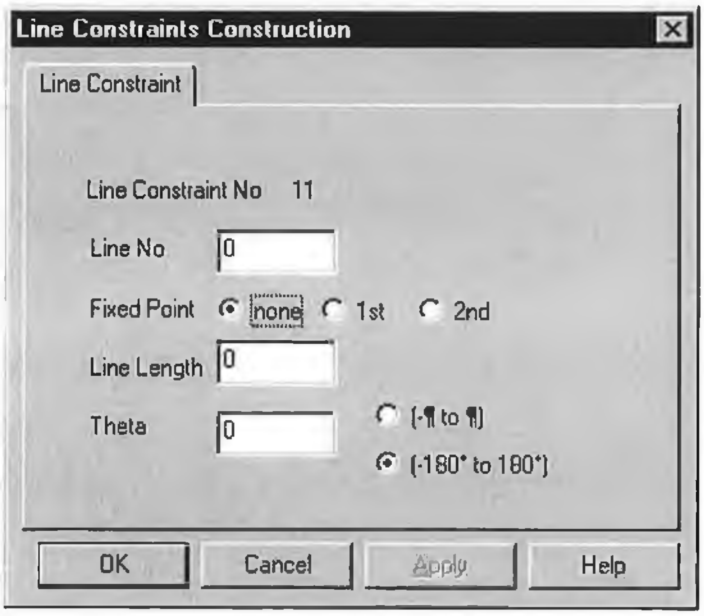
Radius 0 Radial iNode 0

OK | Cancel | Apply | Help

Figure B-8: Circle geometry construction property page

*Line Constraint*

Line constraint construction request starts with issuing command message *ID_LINECONSTRAINTS_CREATE*. The message is capture by CMSSPreDoc class and handled by its member function, OnGeometryLineconstraintsCreate. The present line constraint features are very limited (which only served the requirement of the author for the time being) and not smart enough. Property sheet is used for handling interface instead of a dialog box. This gives space for future expansion. The property page controlled by CPPgLnCnt class under CPShGen property sheet. Figure B-9 presents the display of the line construction property page.


Line Constraints Construction

Line Constraint

Line Constraint No 11

Line No 0

Fixed Point [✓ none] 1st 2nd

Line Length 0

Theta 0 (-¶ to ¶) [✓ (-180° to 180°)]

Figure B-9: Line Constraint construction property page

### a.II Edition

All geometry entity attributes can be edited. One needs not to reconstruct the whole entity if a simple mistake made during construction. Two different selection interface methods have been incorporated to achieve this objective. One is by using mouse to point the entity and then issues edit command message. The other, by menu command message, where a user select a menu item which is to edit an entity and entity reference number is required.

*Point*

The first type of edition method is incorporated for point edition. To select a point to edit, move the mouse near to the target point, then click on the right button of the mouse. By clicking the right mouse button, a window message, *WM_RBUTTONUP* is fired and captured by CMSSPreView class and routed to its member function, OnRButtonUP. The window message carries two parameter, which one of them is the location of the mouse pointer. The below pop-up menu will be created. The pop-up menu coding has been shown previously in

Coding B-3.

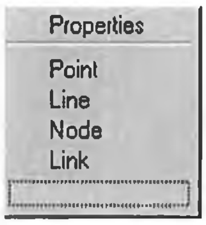


Figure B-10: MSSPre right mouse button click pop-up menu

Select *Point* in the pop-up menu will fire *ID_RBUTTONUP_POINT*. Immediately after the selection, the menu will be destroyed and the mouse point position will be stored in a CMSSPreView member variable.

The *ID_RBUTTONUP_POINT* command message will be captured by CMSSPreView and routed to its member function, *OnRbuttonupPoint*. The handling function will search the nearest point available. Attributes of the nearest point will be extracted from MSSPreDoc. An instance of dialog box handling class, CDlgPointCt, will be created, and the corresponding dialog box will then be displayed. The dialog box is the same used in point construction, which is shown in the left hand side of Figure B-4. All the present attributes of the point are placed in the dialog box before hand. The user now has the opportunity to change the attributes' value. Upon completion of the dialog box, all data in the dialog box will be transferred to the interface handling class and further transferred to CMSSPreView. The data in MSSPreDoc will be updated accordingly. The message and data flow can be generalised as the diagram in Figure B-11.

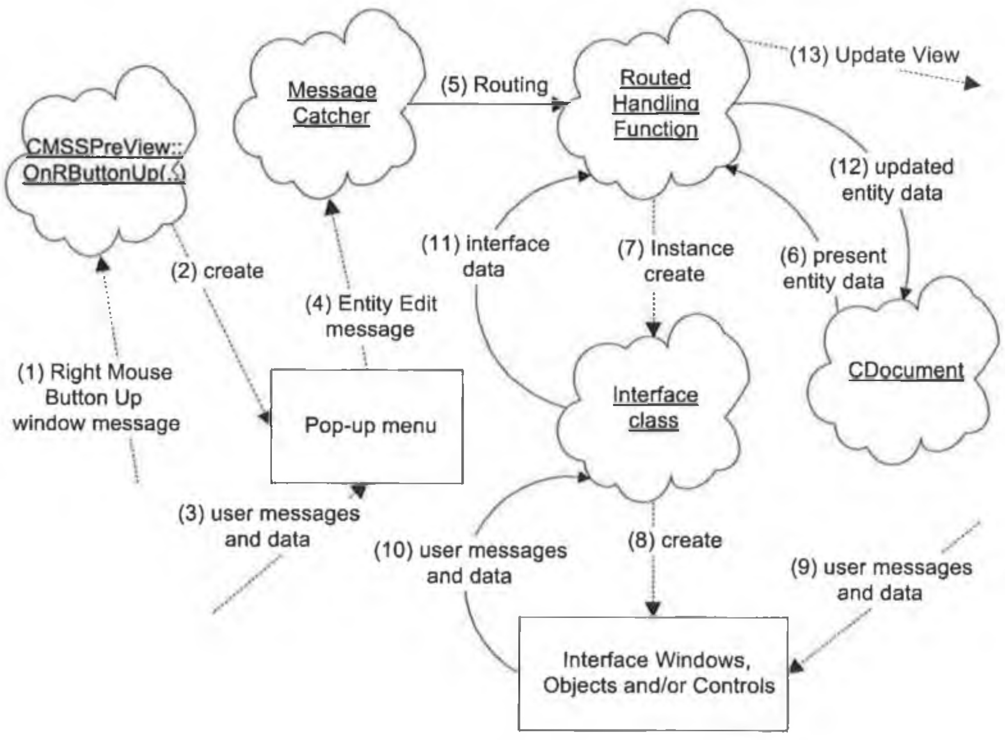
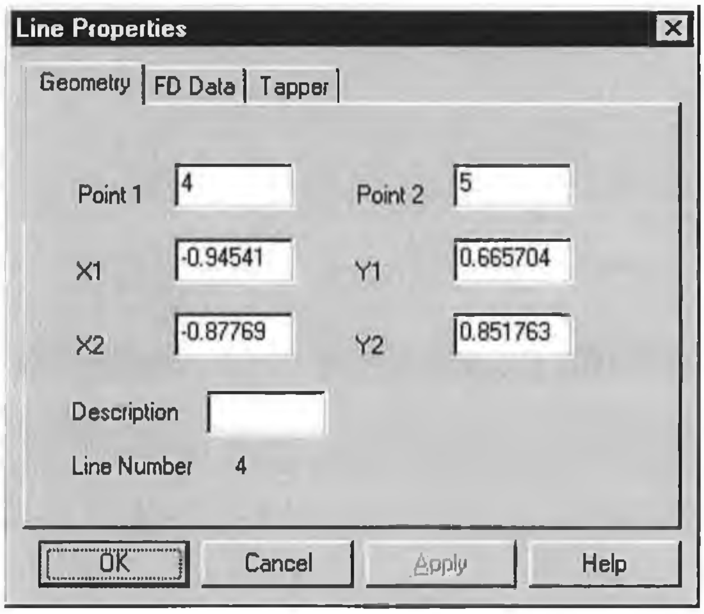


Figure B-11: MSSPre right mouse button click message and data flow diagram

*Line*

Line edition uses the similar method as the point. The selection of *Line* at the pop-up menu fires *ID_RBUTTONUP_LINE* and routed to CMSSPreView::OnRbuttonupLine function. The member functions creates an instance for CPShGen class as the interface handling class which, binds three property page class, namely, CPPgLnGeo, CPPgLNFD and CPPgTapper. The first property page is used to edit the geometry attributes of the line. The other two are for modelling parameters setting, and will be shown in a later section. The line geometry edition property page display is shown as the following:



Point 1: 4
Point 2: 5
X1: -0.94541
Y1: 0.665704
X2: -0.87769
Y2: 0.851763
Description:
Line Number 4

Figure B-12: Line edition property page

*Arc*

Arc edition uses different method compared to line and point. The message and data flow of geometry edition process can be generalized as shown in Figure B-13. Two separate interfaces are required. A user has to select the entity from a list to edit with the Selection Interface, which controls by the *Selection Interface Class*. With the selected entity reference, all corresponding attributes will be transferred from CDocument class to the handling function. The attributes are further transferring to the Edition Interface, which is controlled by *Edition Interface Class*. The user will then make whatever amendment. Updated data will be sent back to handling function via the interface class. The data in the CDocument will be updated accordingly in the handling function. The window display will be updated to reflect the changes.

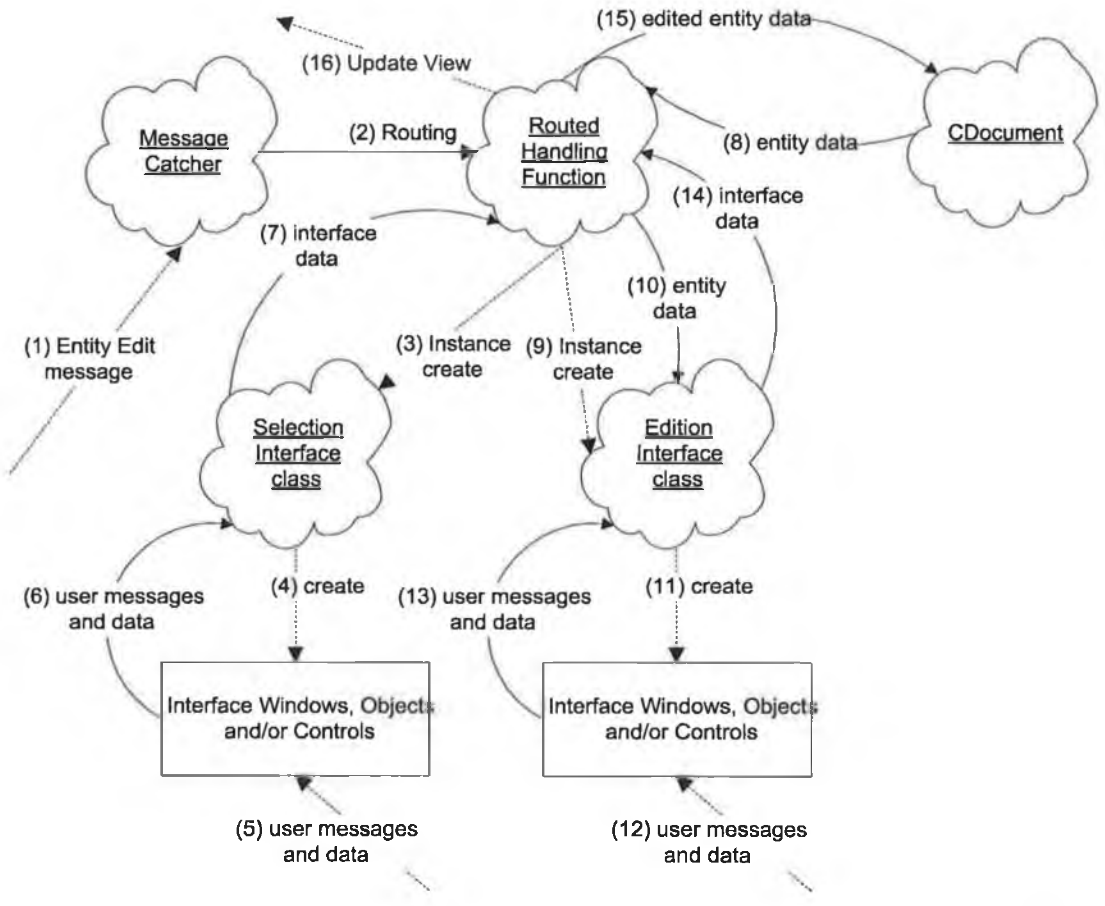
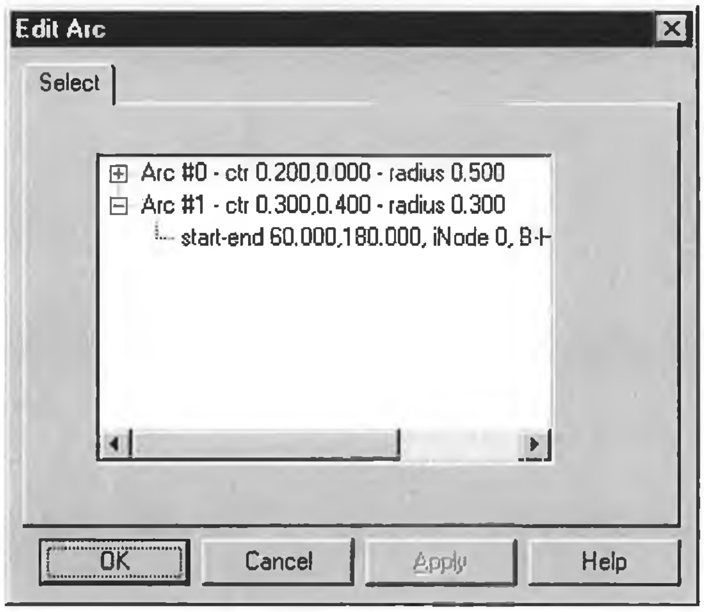


Figure B-13: Geometry edition message and data flow diagram

Arc edit begins with *ID_GEOMETRY_EDIT_ARC*, which is fired by selecting *Geometry – Edit – Arc* in the main menu. CPShGen class is used as *Selection Interface class* and it consists of a property page, which is controlled by CPPgSelect. Figure B-14 shows the property page of the selection interface. After completion of the selection, the edit interface property sheet will be appeared. The edit property sheet is the same as the construction property sheet. The data in the document class will be updated accordingly upon the destruction of the edit property sheet.



```
⊞ Arc #0 - ctr 0.200,0.000 - radius 0.500
⊟ Arc #1 - ctr 0.300,0.400 - radius 0.300
   └ start-end 60.000,180.000, iNode 0, B-H
```

Figure B-14: Arc selection property page

*Circle*

Editing a circle is pretty much similar to editing an arc. The message and data flow is the same as shown in Figure B-13. The same selection interface as in Figure B-14 with the same messages capturing as the arc. CMSSPreView::OnGeometryEditCircle is the handling function for the circle edition and the command message is *ID_GEOMETRY_EDIT_CIRCLE*. The edition interface is handled by CPshGen property sheet class which binds two property pages, namely CPPgCirGeo class and CPPgLNFD class. The former class is used to edit the circle geometry attributes, while the latter is used to alter the modelling parameters. This is exactly the same as the circle construction interface set-up.

*Line Constraint*

Line constraint edition uses the second method, where the user has to select from the main menu. The whole implementation is very much similar to arc and circle edition, just different message ID and different handling class. The message ID is *ID_GEOMETRY_LINECONSTRAINTS_EDIT*, captured by MSSPreDoc class and handled by *MSSPreDoc:::OnGeometryLineconstraintsEdit*. The selection interface is the same as the arc and the circle editions. Whereas the edition interface is the same as its construction interface, as shown in Figure B-9

### a.III Deletion

All the geometry entities deletion shares the common method, but minor differences in implementation. Figure B-15 shows the general message and data flow diagram of MSS pre-processor geometry entity deletion.

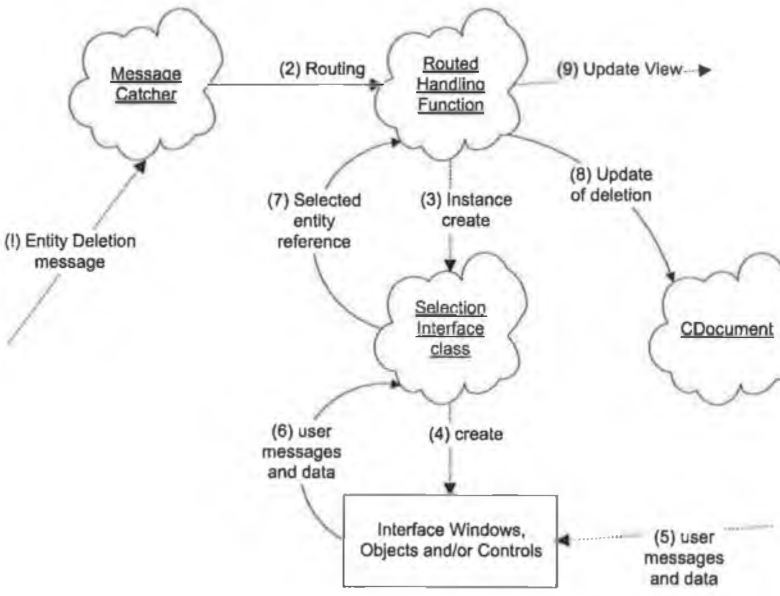

Message Catcher
(2) Routing
Routed Handling Function
(9) Update View
(1) Entity Deletion message
(7) Selected entity reference
(3) Instance create
(8) Update of deletion
Selection Interface class
CDocument
(6) user messages and data
(4) create
Interface Windows, Objects and/or Controls
(5) user messages and data

Figure B-15: Geometry entity deletion message and data flow diagram

All geometry entity deletions use the property page as shown in Figure B-14 for entity select interface. The following table shows message ID and handling class for different entities deletion.

Table B-2:Command message ID and handling function for different entity deletion

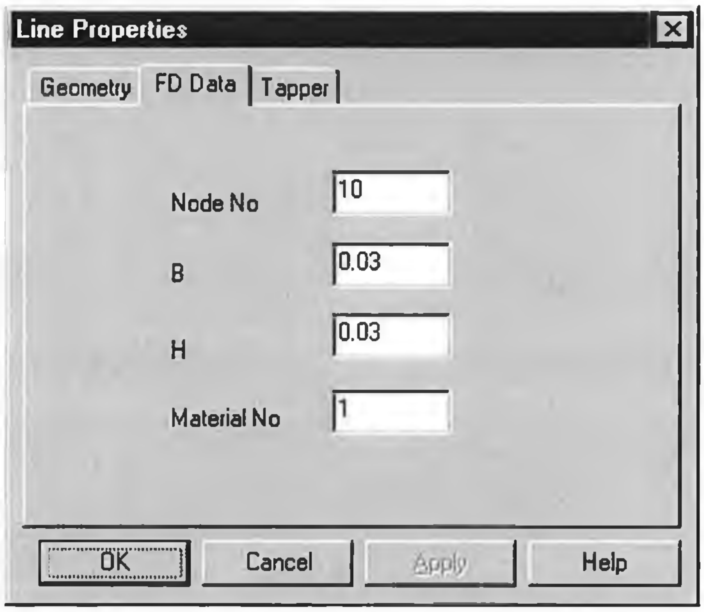
| Entity | Command Message ID | MessageCatcher::Handling Function |
|---|---|---|
| Point & Line | ID_GEOMETRY_DELETE_ POINT | CMSSPreView::OnGeometryDelete Point |
| Arc | ID_GEOMETRY_DELETE_ ARC | CMSSPreView::OnGeometryDelete Arc |
| Circle | ID_GEOMETRY_DELETE_ CIRCLE | CMSSPreView::OnGeometryDelete Circle |
| Line Constraint | ID_GEOMETRY_ LINECONSTRAINTS_DELETE | CMSSPreDoc::OnGeometry LineconstraintsDelete |

## b. Applying Line Constraints

Applying line constraints command message is required to start considering all the constructed line constraints and working on appropriate adjustments. The command message ID is *ID_GEOMETRY_APPLYCONSTRAINTS* and is handled by *OnGeometryApplyconstraints*, a member function of CMSSPreDoc class.

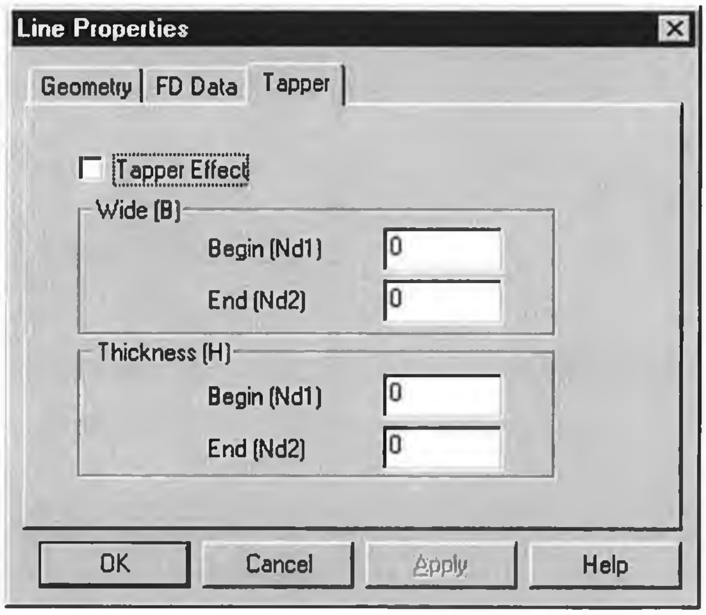
## c. Modelling Parameters of Geometry Entities

All geometry entities' modelling parameters can be set while editing the entities. Modelling parameters property page class, CPPgLNFD, is attached under the property sheet. Figure B-16 shows the property page display. One will have to provide the number of nodes to be generated on the entity, cross section dimension and material model reference. Taper effects are included in MSS pre-processor. Along with the property sheet, CPPgTapper class, which handles tapering property page (Figure B-17), is bind.



| | |
|---|---|
| Node No | 10 |
| B | 0.03 |
| H | 0.03 |
| Material No | 1 |

Figure B-16: Modelling parameters property page



[ ] Tapper Effect

Wide (B)

| | |
|---|---|
| Begin (Nd1) | 0 |
| End (Nd2) | 0 |

Thickness (H)

| | |
|---|---|
| Begin (Nd1) | 0 |
| End (Nd2) | 0 |

Figure B-17: Tapering property page

### d. Workspace Variables

3 pairs of variables are used to define the display window of the workspace. They are called x-y scale factors, x-y origins, and x-y ranges. Figure B-18 shows the definition of the second and the last pair of variables. Both pairs of variables are expressed in device unit (which is the pixel in this case). x-y ranges give the extents of the whole window boundary in horizontal and vertical. Whereas, x-y origins define the location of the logical origin position in term of device unit from the edge of window boundary. Note, the y-origin calculated from the upper boundary of the window. The scaling factors are used to relate the real location to the display location. For example, a circle with radius of 0.1m with 100 in x and y scale factors will turn up to be a circle with 10 pixels in radius on the window. The original direction of y-axis is from top to bottom. To swap the direction, one simply provides a negative value in the y scale factor.

Figure B-3 can be used to describe the message and data flow of setting the workspace variables. The command message ID is *ID_GEOMETRY_WORKSPACEVAR*, catch by MSSPreView class and handled by MSSPreView::OnGeometryWorkspacevar. The interface handling class is CPShGen containing CPPgWSVar property page. Property page is shown in Figure B-19.

### e. Entities Display Settings

A user can choose to display certain types of entity on the screen and mask off the rest. Under present development, six groups are available to be selected, they are point numbers, entity numbers, nodes, links, node numbers, and link numbers. Figure B-20 shows all the display of the above mentioned entity displays. Table B-3 shows message IDs and handling functions of the respective display settings. When a message is received, the handling function will update respective entity display flag. While drawing the window, flags must be checked before displaying any respective entity on the screen. The detail implementation of drawing the window is placed in the CMSSPreView::OnDraw function. To disable a display option, simply uncheck respective menu item.

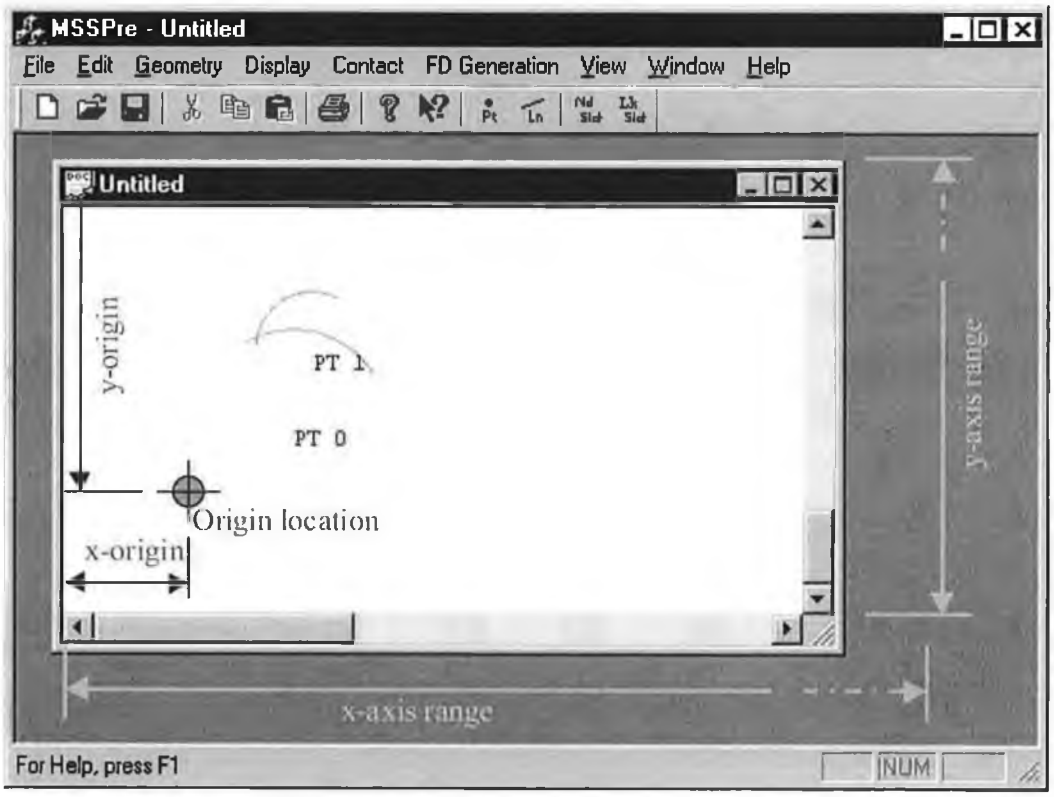
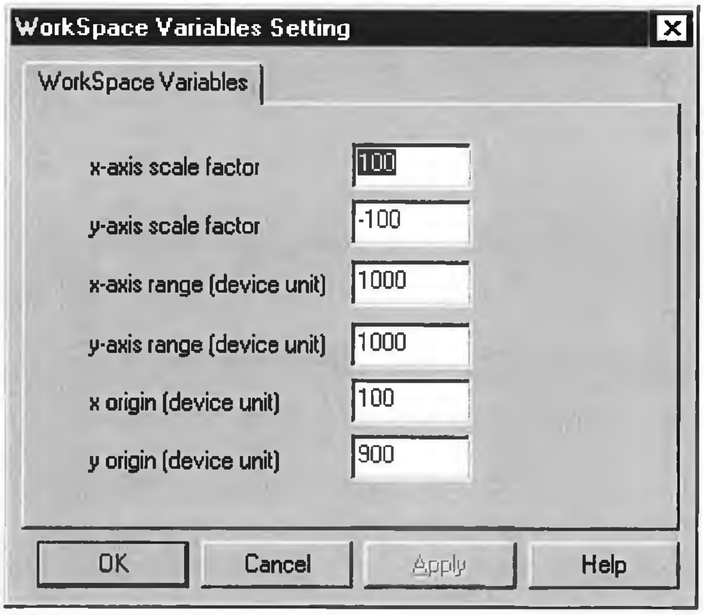


Figure B-18: Workspace variable definitions



WorkSpace Variables Setting

WorkSpace Variables

| | |
|---|---|
| x-axis scale factor | 100 |
| y-axis scale factor | -100 |
| x-axis range (device unit) | 1000 |
| y-axis range (device unit) | 1000 |
| x origin (device unit) | 100 |
| y origin (device unit) | 900 |

Figure B-19: Workspace variables property page

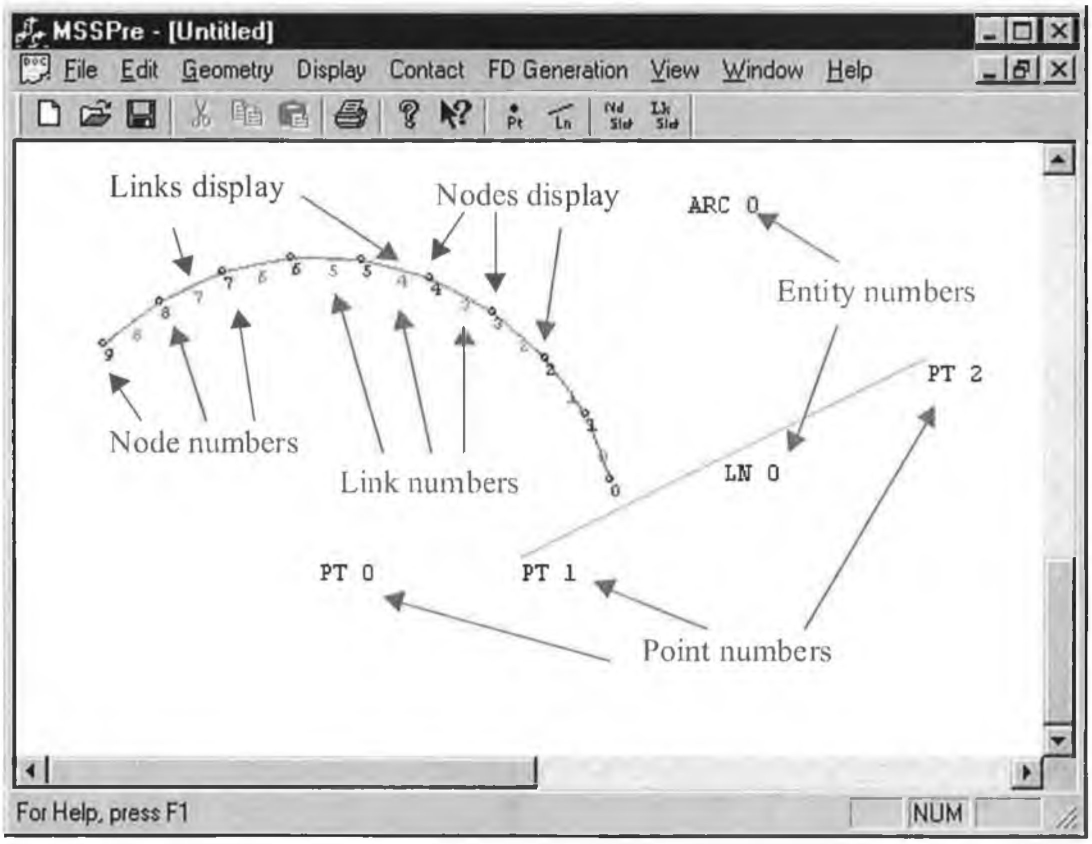


Figure B-20: Entities display representation

Table B-3:Command message ID and handling function for different entity display settings

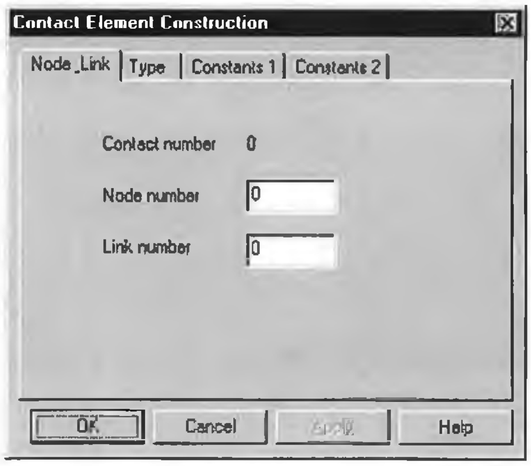
| Entity | Command Message ID | MessageCatcher::Handling Function |
|---|---|---|
| Point No | ID_DISPLAY_POINTNO | CMSSPreView::OnDisplayPointno |
| Line No | ID_DISPLAY_LINENO | CMSSPreView::OnDisplayLineno |
| Nodes | ID_DISPLAY_NODES | CMSSPreView::OnDisplayNodes |
| Links | ID_DISPLAY_LINKS | CMSSPreView::OnDisplayLinks |
| Node No | ID_DISPLAY_NODENO | CMSSPreView::OnDisplayNodeNo |
| Link No | ID_DISPLAY_LinkNO | CMSSPreView::OnDisplayLinkNo |

## f. Contact Element

General contact element is deal under *Contact – Contact Element* in the main menu. The entity is maintained with 3 main functions, which are create, edit, and delete.

MSS pre-processor provides four different methods in helping to define contact elements. The first method is single entry method, where a user will have to provide the contact node and contact surface of the contact element. Second method is called multi nodes – single surface selection. A series of nodes are selected by using mouse pointer and left mouse button. Hold the left mouse button and move the mouse to the desired area to form a box, where all nodes inside the box will be selected. Then the corresponding contact surface is provided by giving the *link* reference number. The third method is the reverse of the second method, where multiple links are selected through interface of mouse capturing, and single contact node reference is provided. The last method is called, multiple selections, where a user can gives multiple nodes and links at the same time and all possible matches will be used to construct respective contact element.

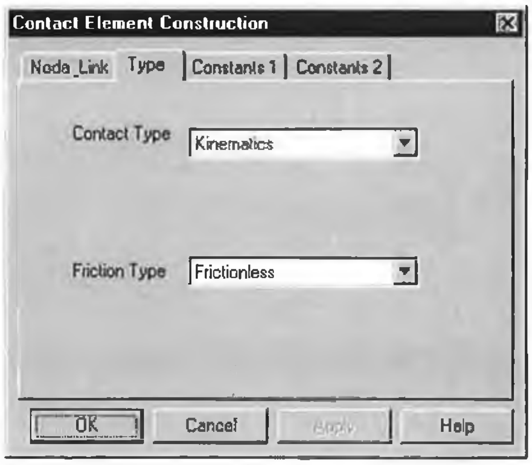
The first and the last methods are very similar. Figure B-3 can be used to describe message and data flow of the methods implementation. The interface class consists of four property pages under CPShGen class, a property sheet handling class. All property pages but the first are common among the two methods. For single selection method, the property pages are handled by CPPgContNdLk, CPPgContType, CPPgPSContConst, and CPPgPSContConst2 respectively. While multiple selection method uses CPPgMultiNdLkSel as the first property page. All the five property pages are shown in Figure B-21. The other main difference is handling class implementation after the destruction of the data entry property sheet. Additional algorithms are required to create multiple contact elements based on multiple nodes and links entries in multiple selection method.



Contact number 0

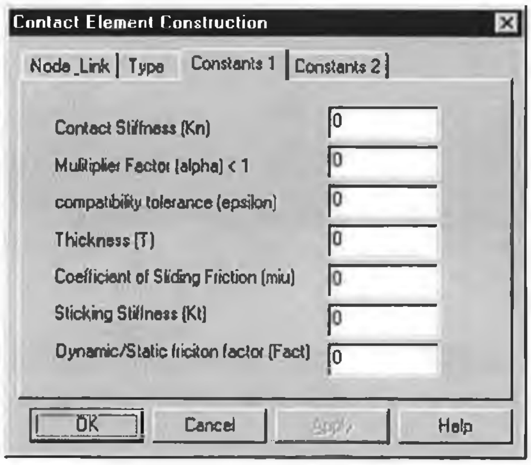
Node number 0

Link number 0



Contact Type: Kinematics

Friction Type: Frictionless



Contact Stiffness (Kn) 0

Multiplier Factor (alpha) < 1 0

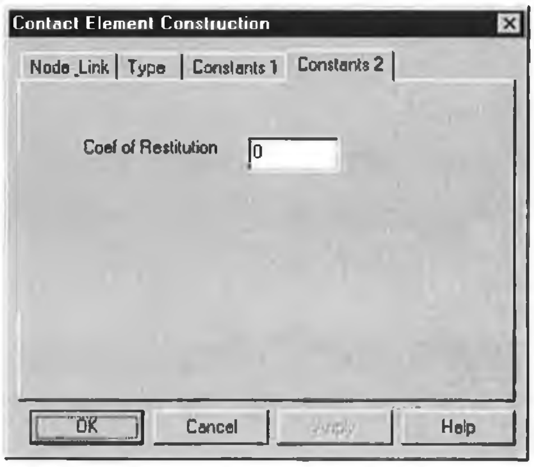
compatibility tolerance (epsilon) 0

Thickness (T) 0

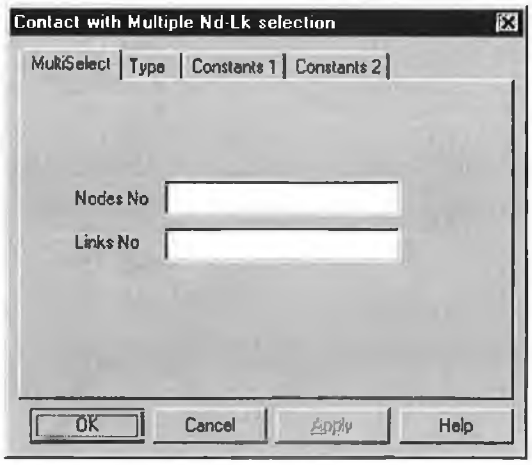
Coefficient of Sliding Friction (miu) 0

Sticking Stiffness (Kt) 0

Dynamic/Static friciton factor (Fact) 0



Coef of Restitution 0



Nodes No

Links No

Figure B-21: Contact element interfaces handled by CPPgContNdLk, CPPgContType, CPPgPSContConst, CPPgPSContConst2 and CPPgMultiNdLkSel (from left to right, top to bottom).

In multiple nodes – single surface and multiple surfaces – single node methods, same interface as single entry method, but the user will only be allowed to enter either one node reference for multiple surfaces selection method or one surface reference for multiple nodes selection method. The front-end message and data flow is shown below:

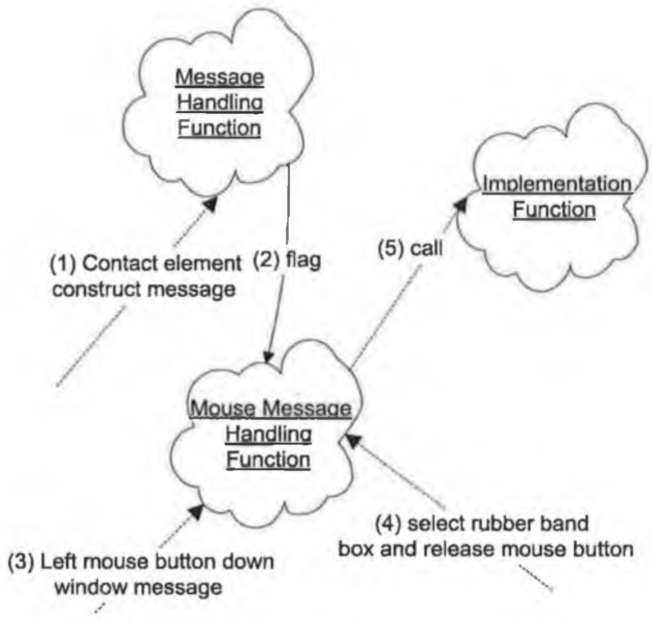

Message Handling Function
Implementation Function
(1) Contact element construct message
(2) flag
(5) call
Mouse Message Handling Function
(4) select rubber band box and release mouse button
(3) Left mouse button down window message

Figure B-22: Front-end message and data flow diagram of multiple nodes or surfaces contact element construction

The element construction command messages are :

- *ID_CONTACT_POINTSURFACE_SELNODES*
- *ID_CONTACT_POINTSURFACE_SELLINKS*

They are for multi-nodes selection and multi-links selection respectively. The message will be captured by MSSPreView class and routed to respective handling functions. In the implementation of the functions, selection flag will be set accordingly. The system is readied to accept mouse selection. The user will mobilize the mouse to get a *rubber band* box. When pressing the left mouse button, a window message is fired, namely, WM_LBUTTONDON. Preparation for *rubber band* box construction is made by the mouse message handling class. By holding down the left mouse button and move the mouse pointer to the desired destination and release, a *rubber band* box will be formed. After releasing of the mouse button, area of selection will be obtained. Validation is made to find all the nodes/links within the selected area. It is carried out in the implementation function. The back-end message and data flows are connected to the implementation function. The whole back-end process is similar to that mentioned in the first and the last methods of contact element construction.

Edit and delete implementations are very much similar to the other entities.

## g. Revolute Joint Element

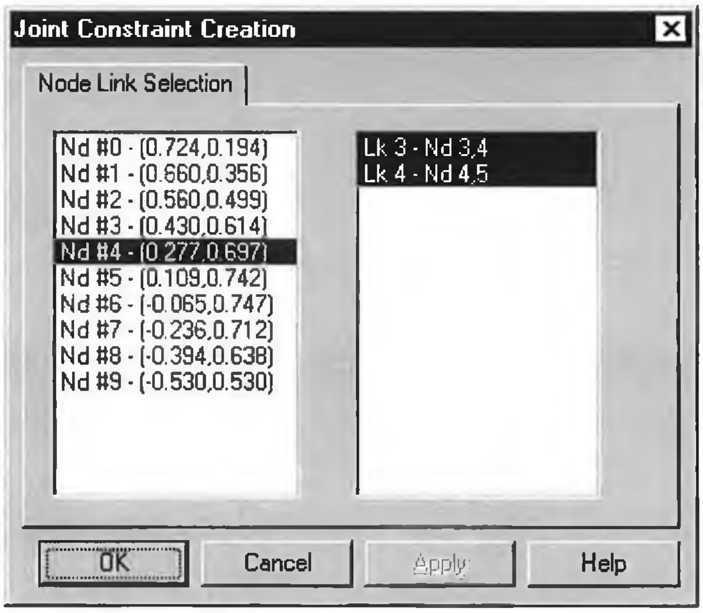
All Revolute joint element functions can be found under *Contact – Joint Constraint* in the main menu. They are create, edit and delete.

The message and data flow is very similar to the one shown in Figure B-13. An additional interface will be shown which is controlled by CPPgNdLk class under CPshGen property sheet class (named as selection interface class). All the nodes and its associate links will be listed as shown in Figure B-23. The user is required to select one node and two of the node-associated links to form the revolute joint. The routed handling function will further create another interface, as shown in Figure B-24.

Edit and delete implementations are very much similar to the other entities.

## h. Node-Link Generation

Node-link generation is similar to meshing processing, where geometry entities are discretised into links by nodes. The discretisation is depending on the number of nodes set (one of the modelling parameters). The command message is *ID_FDGENERATE_NODELINK* and handled by OnGenerateNodelink of CMSSPreDoc class. The detail implementation of the node-link generation will not be covered in this thesis.

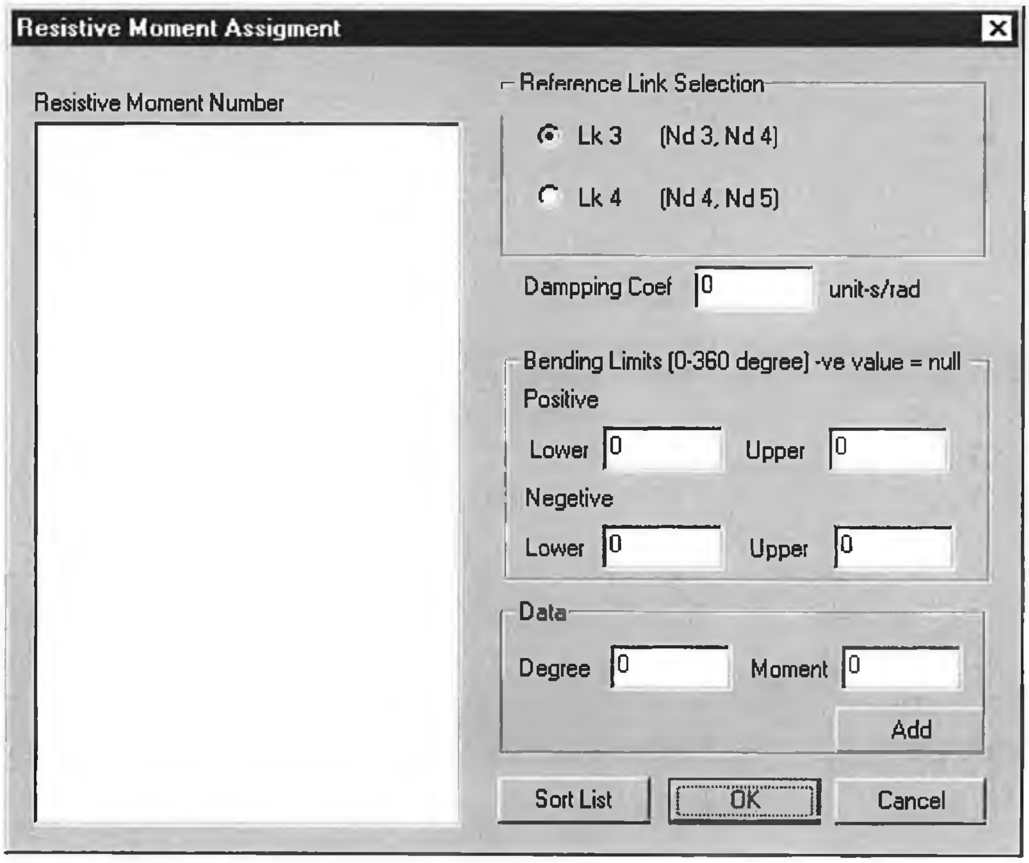

Joint Constraint Creation

Node Link Selection

```
Nd #0 - (0.724,0.194)
Nd #1 - (0.660,0.356)
Nd #2 - (0.560,0.499)
Nd #3 - (0.430,0.614)
Nd #4 - (0.277,0.697)
Nd #5 - (0.109,0.742)
Nd #6 - (-0.065,0.747)
Nd #7 - (-0.236,0.712)
Nd #8 - (-0.394,0.638)
Nd #9 - (-0.530,0.530)
```

```
Lk 3 - Nd 3,4
Lk 4 - Nd 4,5
```

Figure B-23: Node-links selection during revolute joint creation


Resistive Moment Assigment

Resistive Moment Number

Reference Link Selection

(●) Lk 3 (Nd 3, Nd 4)

( ) Lk 4 (Nd 4, Nd 5)

Dampping Coef 0 unit-s/rad

Bending Limits (0-360 degree) -ve value = null

Positive

Lower 0 Upper 0

Negetive

Lower 0 Upper 0

Data

Degree 0 Moment 0

Figure B-24: Revolute joint element properties dialog box

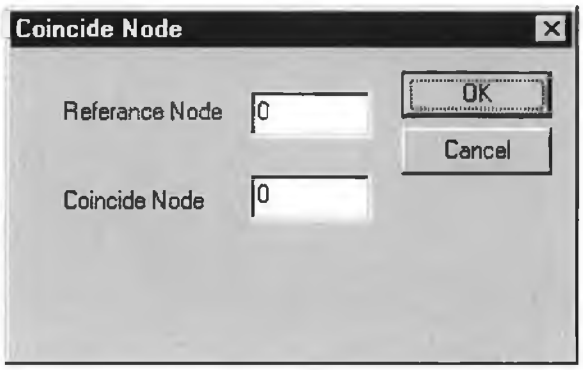
### i. Coincide nodes

After node-link generation, coincide nodes may be generated especially at connections of geometry entities. Three ways of eliminating coincide nodes are provided by MSSPre, namely Automatic, Selection, and Dialog box. Their respective message IDs and handling classes are the following:

Table B-4: Coincide nodes elimination message IDs and handling functions

| Method | Command Message ID | Handling Function |
|---|---|---|
| Automatic | ID_FDGENERATION_COINCID ENODES | CMSSPreDoc::OnFdgenerationCoin cidenodes |
| Selection | ID_FDGENERATION_NODESR EDUNDANCY | CMSSPreView::OnFdgenerationNod esredundancy |
| Dialog Box | ID_FDGENERATION_COINCID ENODEDLG | CMSSPreDoc::OnFdgenerationCoin cidenodedlg |

Automatic coincided nodes elimination will clear all nodes but one having near coordinate location. The near is referring to 0.0001 unit of radius. This constant can be altered. Selection method is similar to multiple nodes/links selection in contact element construction. Whereas the last method implements the normal way via dialog box interface. It follows the message and data flow of Figure B-3. The reference nodes and coincide nodes have to be provided in a dialog box, as shown in Figure B-25.



Referance Node 0

Coincide Node 0

Figure B-25: Coincide node elimination dialog box

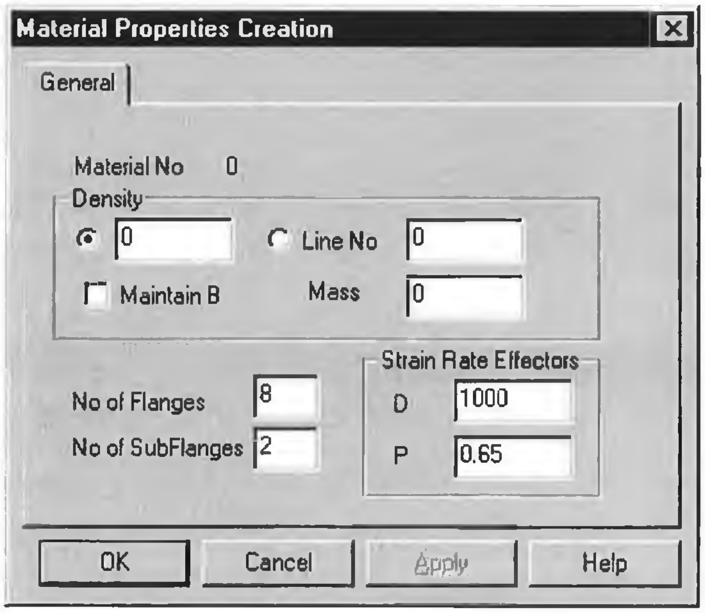
## j. Material Model

Material models are maintained under *FD Generation – Material Prop* menu. Four functions are available, they are create, edit, copy and delete. The implementations of create, edit and delete functions are the same as the geometry entity create, edit and delete. The copy function copies a source material attributes to a new material model. Figure B-26 and Figure B-27 are the interfaces to define material properties.

MSS pre-processor provides two ways to define material density. The first is the normal entry of density value in the density edit box of the property page shown in Figure B-26. The other way is providing the *line* reference number and the desired mass value of the *line*. The density will be calculated automatically. The cross section dimension (either the base, B, or the height, H) of the line can be altered according, with reference to the specified density and the mass required. The height will be altered automatically if the *Maintain B* check box of Figure B-26 is selected. Alternative interface is provided to reduce confusion. It is under *FD Generation – Assign Material* of the main menu. The interface display is shown in Figure B-28, where a user can select whether to maintain the base or the height of the cross section. Table B-5 shows the message IDs and handling functions of respective material model functions.

Table B-5: Material model message IDs and handling functions

| Method | Command Message ID | Handling Function |
|---|---|---|
| Create | ID_FDGENERATION_MATERIALPROP_CREATE | CMSSPreDoc::OnFdgenerationMaterialpropCreate |
| Edit | ID_FDGENERATION_MATERIALPROP_EDIT | CMSSPreDoc::OnFdgenerationMaterialpropEdit |
| Copy | ID_FDGENERATION_MATERIALPROP_COPY | CMSSPreDoc::OnFdgenerationMaterialpropCOPY |
| Delete | ID_FDGENERATION_MATERIALPROP_DELETE | CMSSPreDoc::OnFdgenerationMaterialpropDelete |
| Assign | ID_FDGENERATION_ASSIGNMATERIAL | CMSSPreDoc::OnFdgenerationAssignmaterial |



Material No 0

Density: (●) 0 ( ) Line No 0

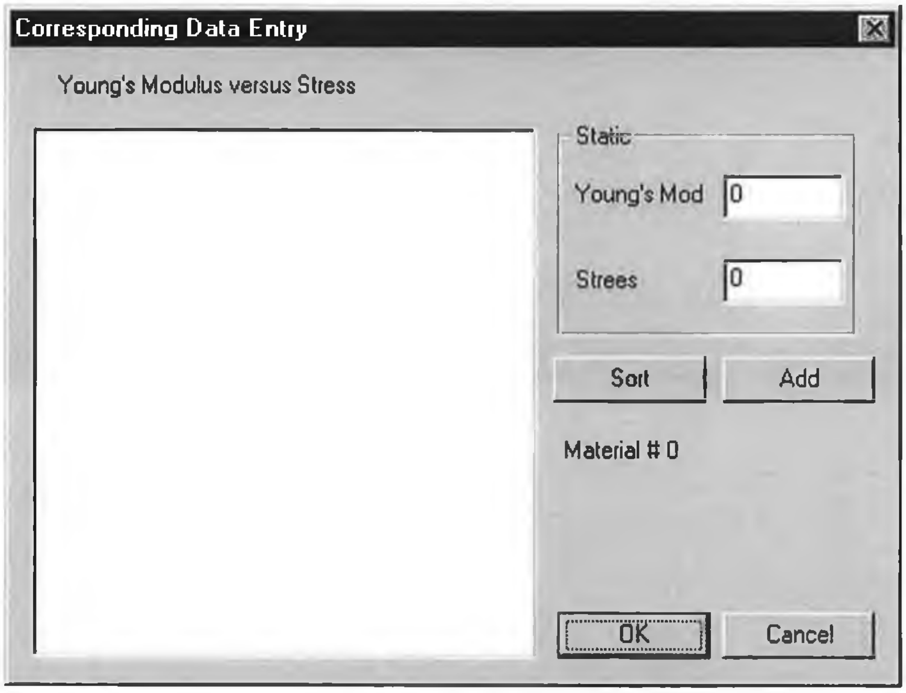
[ ] Maintain B  Mass 0

No of Flanges 8

No of SubFlanges 2

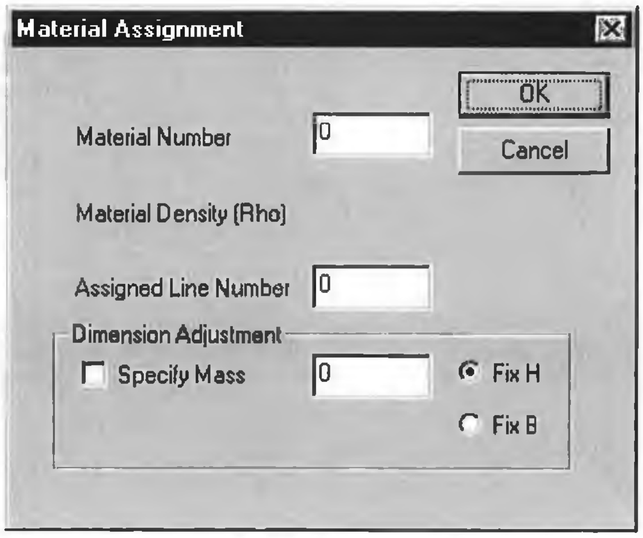
Strain Rate Effectors: D 1000, P 0.65

Figure B-26: Material general attributes property page



Young's Mod: 0
Strees: 0
Material # 0

Figure B-27: Stress-strain relation dialog box



Material Number: 0
Assigned Line Number: 0
☐ Specify Mass: 0
[✓ Fix H] Fix B

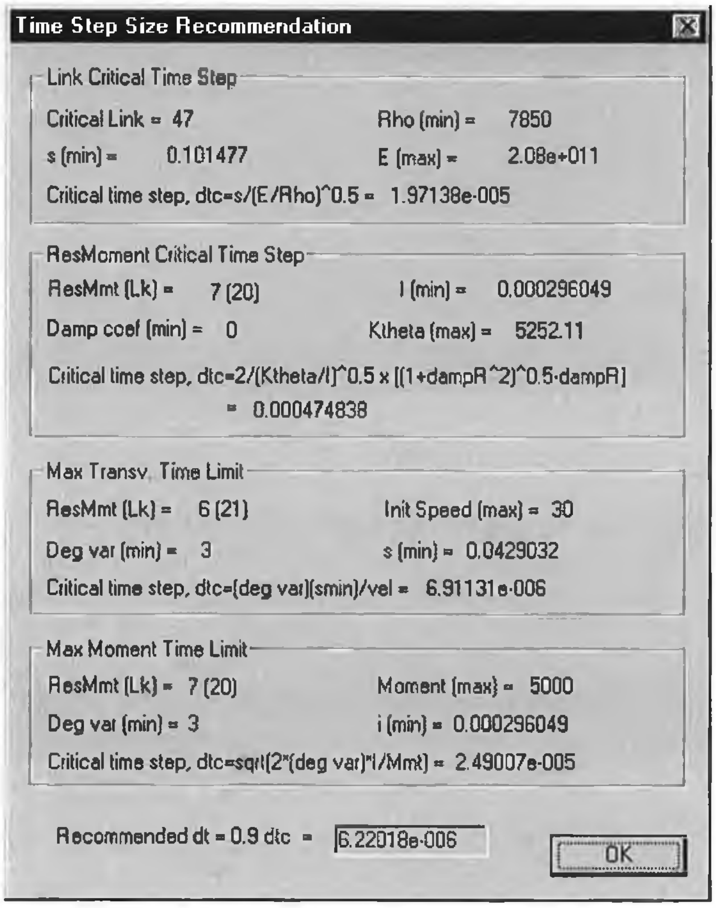
Figure B-28: Assign Material dialog box

## k. Time Step Size Recommendation

MSS pre-processor recommends time step size to a user for the processing of the model in MSS. Command message *ID_FDGENERATION_CRTITCALTIMESTEP* will pop-up an information dialog box as shown in Figure B-29. All critical time step sizes based on different critical criterions are calculated and displayed on the information box. Not all critical criterions are valid in nature, some are still at experimental stage. Refer next chapter for more details.

## l. MSS Processor Code Generation

Before generating the MSS processor text based data file, several overall processing parameters are required. They are the time step size, maximum time step and time step interval. Furthermore, gravitational acceleration effect towards the whole model can be selected. Figure B-30 shows the property page of the pre-generation MSS parameters entry interface. Immediately after the destruction of the property page, MSS pre-processor will create a text-based MSS data file, with "inp" as the extension.


Time Step Size Recommendation

Link Critical Time Step

Critical Link = 47 Rho (min) = 7850

s (min) = 0.101477 E (max) = 2.08e+011

Critical time step, dtc=s/(E/Rho)^0.5 = 1.97138e-005

ResMoment Critical Time Step

ResMmt (Lk) = 7 (20) I (min) = 0.000296049

Damp coef (min) = 0 Ktheta (max) = 5252.11

Critical time step, dtc=2/(Ktheta/I)^0.5 x [(1+dampR^2)^0.5-dampR]
= 0.000474838

Max Transv. Time Limit

ResMmt (Lk) = 6 (21) Init Speed (max) = 30

Deg var (min) = 3 s (min) = 0.0429032

Critical time step, dtc=(deg var)(smin)/vel = 6.91131e-006

Max Moment Time Limit

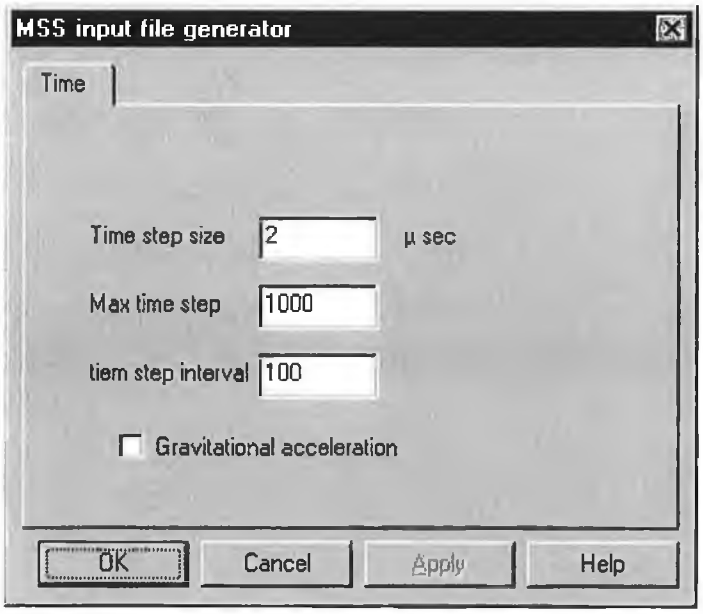
ResMmt (Lk) = 7 (20) Moment (max) = 5000

Deg var (min) = 3 i (min) = 0.000296049

Critical time step, dtc=sqrt(2*(deg var)*I/Mmt) = 2.49007e-005

Recommended dt = 0.9 dtc = 6.22018e-006

Figure B-29: Time step size recommendation information box


MSS input file generator

Time

Time step size 2 μ sec

Max time step 1000

tiem step interval 100

☐ Gravitational acceleration

Figure B-30: MSS processor code pre-generation property page

## B.8 GENERAL DESCRIPTION OF THE MSS PROCESSOR CLASSES

Table B-6 shows the entire list of the classes incorporated into the MSS Processor.

Table B-6: MSS processor classes description

| Class | Type | Description |
|---|---|---|
| CAboutDlg | Interface | Handles of About dialog box |
| CchildFrame | Application | MDI controls |
| CContact | Object | Holds a contact element 's attributes and implementations |
| CEndLk | Object | Holds a end link condition and implementations |
| Clink | Object | Holds a link's attributes and implementations |
| ClkPivot | Object | Holds a link pivot attributes and implementations |
| CMainFrame | Application | Window main frame control |
| CMaterial | Object | Holds a material model attributes and implementations |
| Cmatrix | Base | Matrix formulation and implementations |
| CMSSApp | Application | Initialization class |
| CMSSDoc | Application | Document class |
| CMSSView | Application | View class |
| CNode | Object | Holds a node attributes and implementations |
| CResMmt | Base | Holds a revolute joint base attributes |
| CResMmtPro | Object | Holds a revolute joint attributes and implementations |
| CSPntCnt | Object | Holds a node constraint attributes and implementations |
| CVector | Base | Vector formulation and implementations |

## B.9 MENU DEFINITION OF THE MSS PROCESSOR

The menu system of the MSS Processor is listed as the following coding.

```
IDR_MAINFRAME MENU PRELOAD DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "New",                         ID_FILE_NEW
        MENUITEM "&Open...\tCtrl+O",            ID_FILE_OPEN
        MENUITEM "&Save\tCtrl+S",               ID_FILE_SAVE
        MENUITEM "Save &As...",                 ID_FILE_SAVE_AS
        MENUITEM SEPARATOR
        MENUITEM "&Print...\tCtrl+P",           ID_FILE_PRINT
        MENUITEM "Print Pre&view",              ID_FILE_PRINT_PREVIEW
        MENUITEM "P&rint Setup...",             ID_FILE_PRINT_SETUP
        MENUITEM SEPARATOR
        MENUITEM "Recent File",                 ID_FILE_MRU_FILE1, GRAYED
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                       ID_APP_EXIT
    END
    POPUP "&Edit"
    BEGIN
        MENUITEM "&Undo\tCtrl+Z",               ID_EDIT_UNDO
        MENUITEM SEPARATOR
        MENUITEM "Cu&t\tCtrl+X",                ID_EDIT_CUT
        MENUITEM "&Copy\tCtrl+C",               ID_EDIT_COPY
        MENUITEM "&Paste\tCtrl+V",              ID_EDIT_PASTE
    END
    POPUP "&View"
    BEGIN
        MENUITEM "&Toolbar",                    ID_VIEW_TOOLBAR
        MENUITEM "&Status Bar",                 ID_VIEW_STATUS_BAR
    END
    POPUP "Run"
    BEGIN
        MENUITEM "Start",                       ID_START
        MENUITEM "Stop",                        ID_STOP
        MENUITEM "Status",                      ID_STATUS
    END
    POPUP "&Help"
    BEGIN
        MENUITEM "&About MSS...",               ID_APP_ABOUT
    END
END
```

Coding B-5: MSS processor main menu coding from MSS.rc

## B.10 GENERAL DESCRIPTION OF THE MSS POST-PROCESSOR CLASSES

The post-processor interface classes are shown in the following table.

Table B-7: MSSPost interface classes

| Class | Description |
|---|---|
| CDlgLSMSize | Provide value of *n* in finding least squares velocity and acceleration |
| CDlgTcNdVel | Customize the special request output data file based on a node |
| CDlgTcTime | Customize the special request output data file based on a time step |
| CDlgTrackLk | Customize the special request output data file based on a link |
| CDlgViewOverA | Set display options |
| CFreezeDlg | Display transient response at a particular time step |

## B.11 MENU DEFINITION OF THE MSS POST-PROCESSOR

The menu system of the MSS Post-processor is listed as the following coding.

```
IDR_MAINFRAME MENU PRELOAD DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&New\tCtrl+N",               ID_FILE_NEW
        MENUITEM "&Open...\tCtrl+O",           ID_FILE_OPEN
        MENUITEM "&Save\tCtrl+S",              ID_FILE_SAVE
        MENUITEM "Save &As...",                ID_FILE_SAVE_AS
        MENUITEM SEPARATOR
        MENUITEM "&Print...\tCtrl+P",          ID_FILE_PRINT
        MENUITEM "Print Pre&view",             ID_FILE_PRINT_PREVIEW
        MENUITEM "P&rint Setup...",            ID_FILE_PRINT_SETUP
        MENUITEM SEPARATOR
        MENUITEM "Recent File",                ID_FILE_MRU_FILE1, GRAYED
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                      ID_APP_EXIT
    END
    POPUP "&Edit"
    BEGIN
        MENUITEM "&Undo\tCtrl+Z",              ID_EDIT_UNDO
        MENUITEM SEPARATOR
        MENUITEM "Cu&t\tCtrl+X",                ID_EDIT_CUT
        MENUITEM "&Copy\tCtrl+C",               ID_EDIT_COPY
        MENUITEM "&Paste\tCtrl+V",              ID_EDIT_PASTE
    END
    POPUP "Setting"
    BEGIN
        MENUITEM "LSM Size",                    ID_SETTING_LSMSIZE
    END
    POPUP "&Track"
    BEGIN
        MENUITEM "Node Velocity",               ID_TRACK_NODEVELOCITY
        MENUITEM "Link",                        ID_TRACK_LINK
        MENUITEM "Time",                        ID_TRACK_TIME
    END
    POPUP "&View"
    BEGIN
        MENUITEM "Animate",                     ID_VIEW_ANIMATE
        MENUITEM "Freeze",                      ID_VIEW_FREEZE
        MENUITEM "Overall",                     ID_VIEW_OVERALL
        MENUITEM SEPARATOR
        MENUITEM "Tracked",                     ID_VIEW_TRACKED
        MENUITEM SEPARATOR
        MENUITEM "&Toolbar",                    ID_VIEW_TOOLBAR
    END
    POPUP "&Help"
    BEGIN
        MENUITEM "&About MSSPost...",           ID_APP_ABOUT
    END
END
```

Coding B-6: MSS post-processor main menu coding from MSSPost.rc

## B.12 MSS PROCESSOR OUTPUT DATA FILE SYSTEM

Figure B-31 shows the file system format. The bracket shows the data type and square bracket indicates the reference number of respective entity. The overheads are place at the beginning of the data file. They are the time step size, numbers of nodes and links, and node-link relationships. Then, the initial conditions of the problem are stored, which consist of nodal displacement, velocity, and acceleration, and overall energy levels (consisting the overall kinetic energy, elastic energy and plastic energy) are stored.

The following data is collected in term of time step interval. All the nodal information, linkage information and energies level at a specific time step interval will be stored and followed by next time step interval data. Each time step interval equivalent time steps is stored at beginning of respective time step interval data block. The intervals are sorted in ascending order in the data file, from first interval to the last.

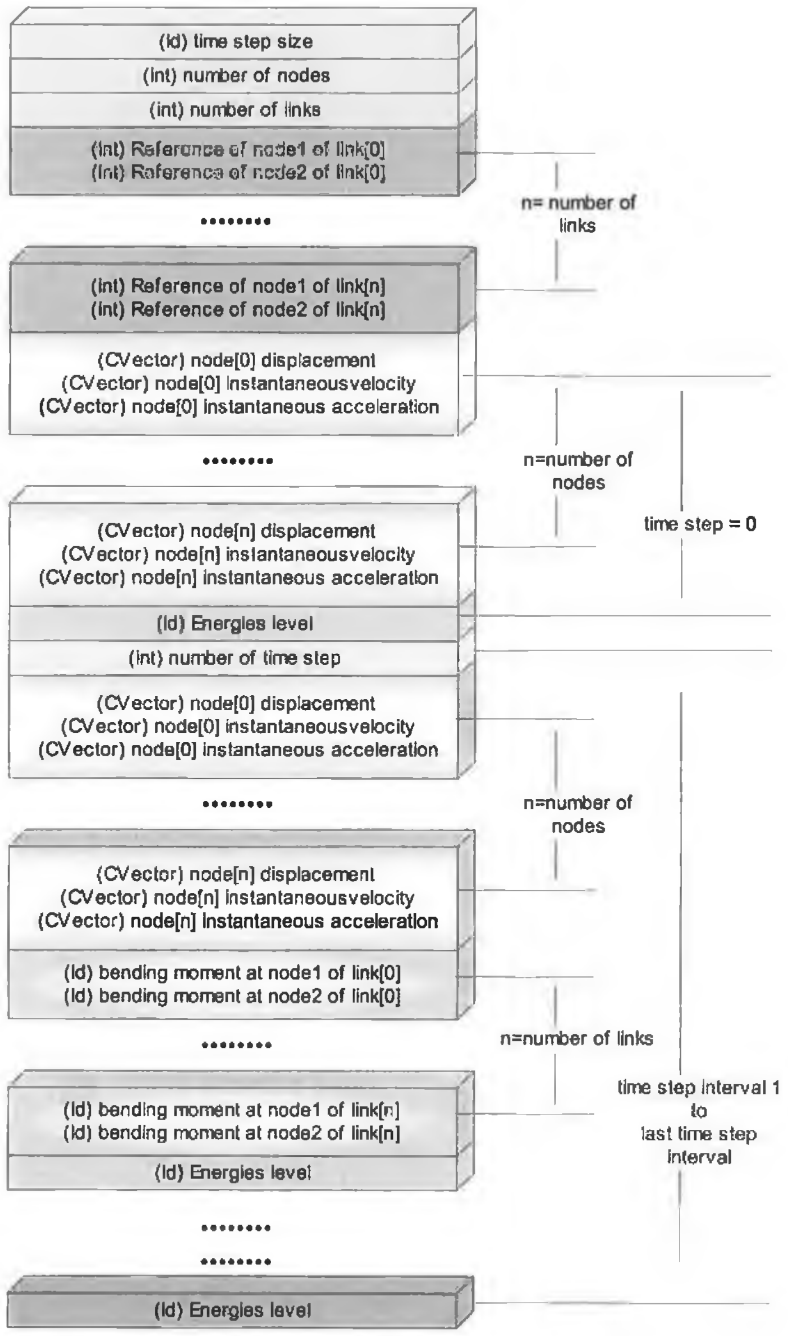

(ld) time step size
(int) number of nodes
(int) number of links
(int) Reference of node1 of link[0]
(int) Reference of node2 of link[0]
n= number of links
(int) Reference of node1 of link[n]
(int) Reference of node2 of link[n]
(CVector) node[0] displacement
(CVector) node[0] instantaneousvelocity
(CVector) node[0] instantaneous acceleration
n=number of nodes
time step = 0
(CVector) node[n] displacement
(CVector) node[n] instantaneousvelocity
(CVector) node[n] instantaneous acceleration
(ld) Energies level
(int) number of time step
(CVector) node[0] displacement
(CVector) node[0] instantaneousvelocity
(CVector) node[0] instantaneous acceleration
n=number of nodes
(CVector) node[n] displacement
(CVector) node[n] instantaneousvelocity
(CVector) node[n] instantaneous acceleration
(ld) bending moment at node1 of link[0]
(ld) bending moment at node2 of link[0]
n=number of links
time step interval 1 to last time step interval
(ld) bending moment at node1 of link[n]
(ld) bending moment at node2 of link[n]
(ld) Energies level
(ld) Energies level

Figure B-31: MSS Processor Output Data File System

# Appendix C: Simulation Results for $2^4$ Factorial Design

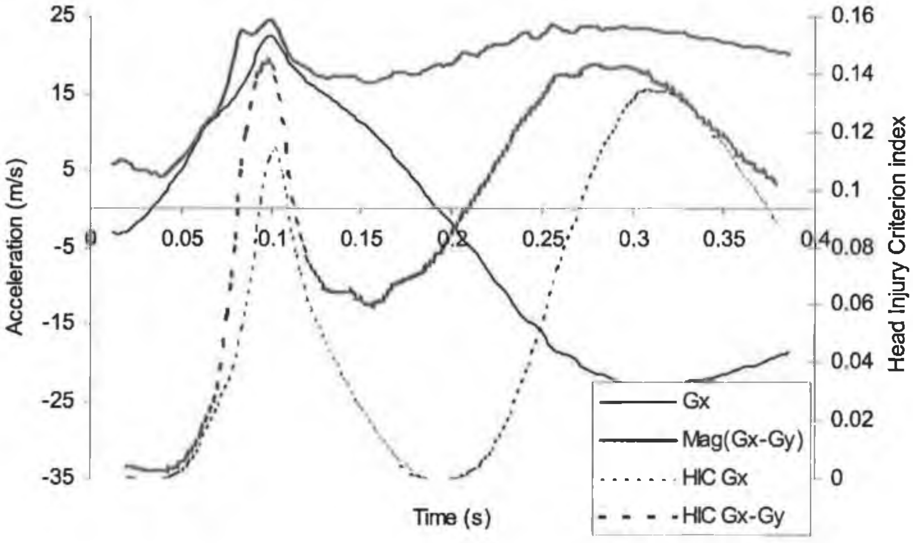


Figure C-1: Node 23 acceleration history of experiment (1)

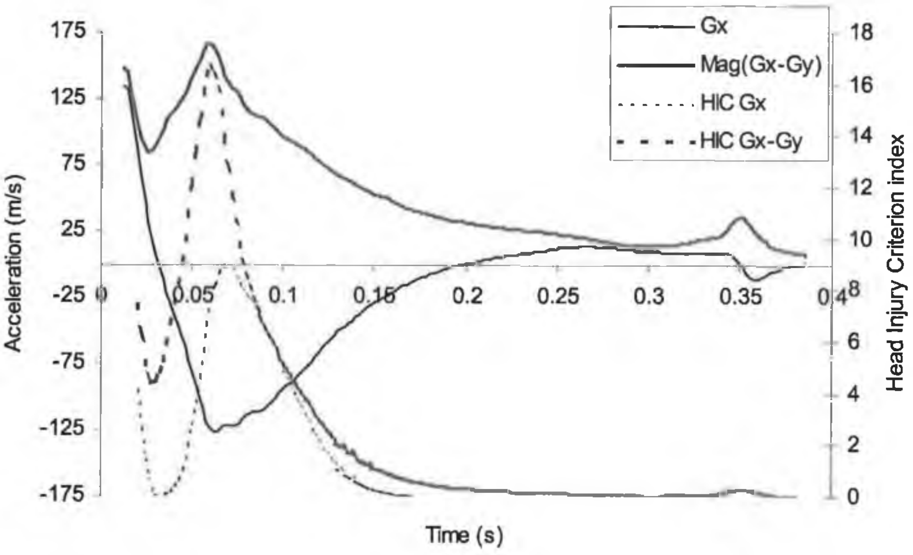


Figure C-2: Node 23 acceleration history of experiment b

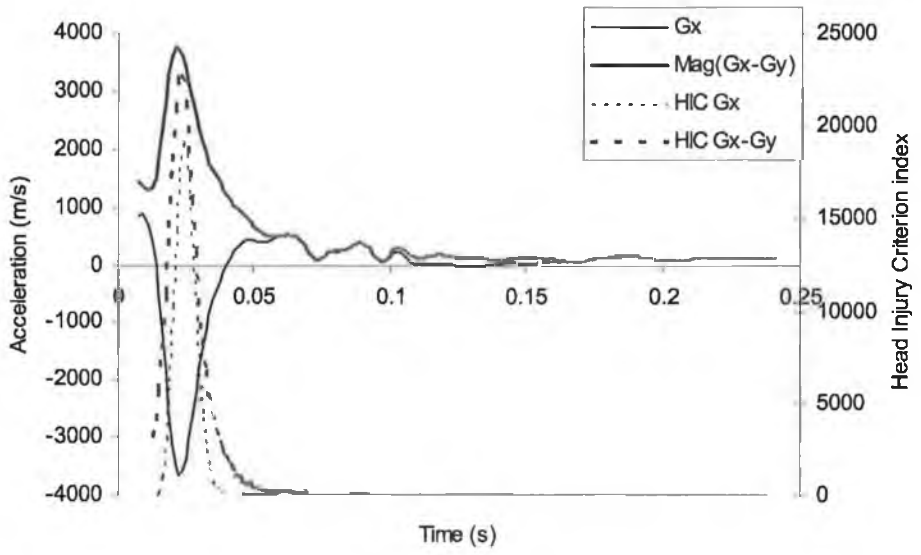


Figure C-3: Node 23 acceleration history of experiment vb

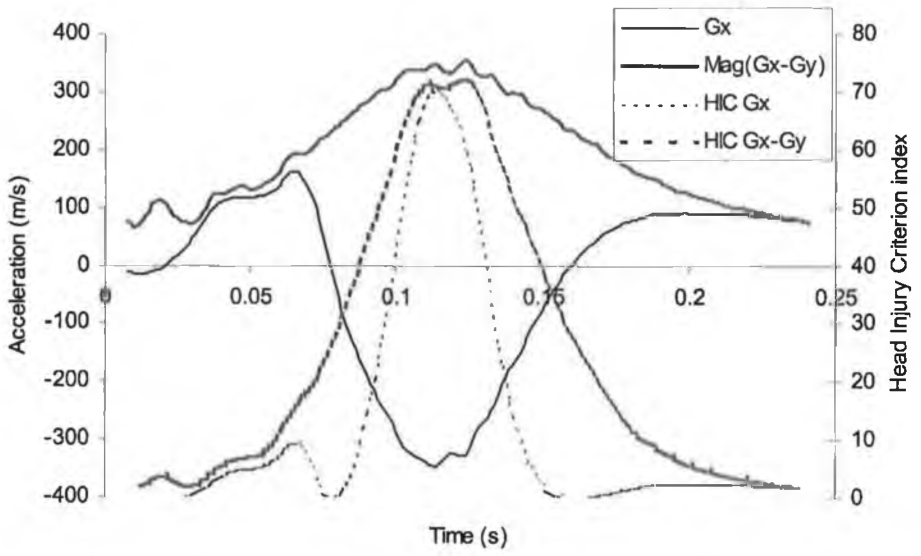


Figure C-4: Node 23 acceleration history of experiment v

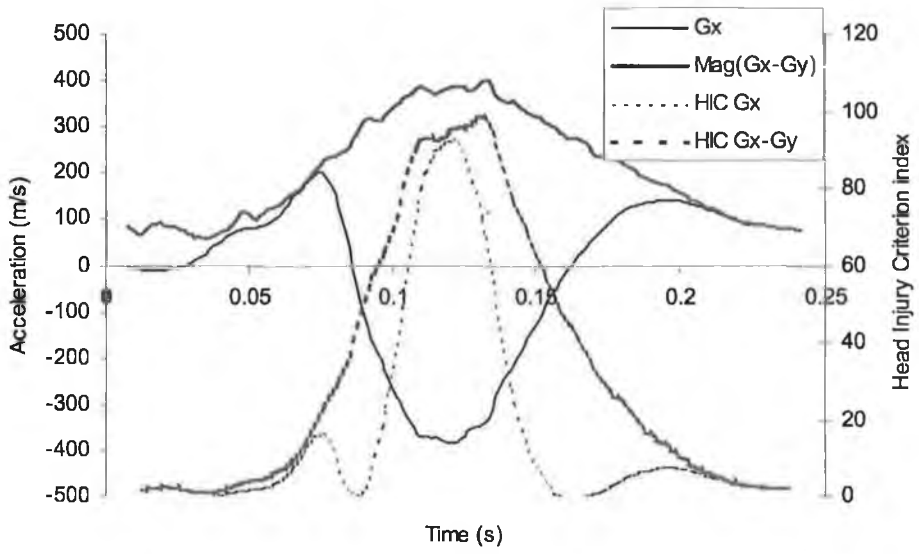


Figure C-5: Node 23 acceleration history of experiment wv

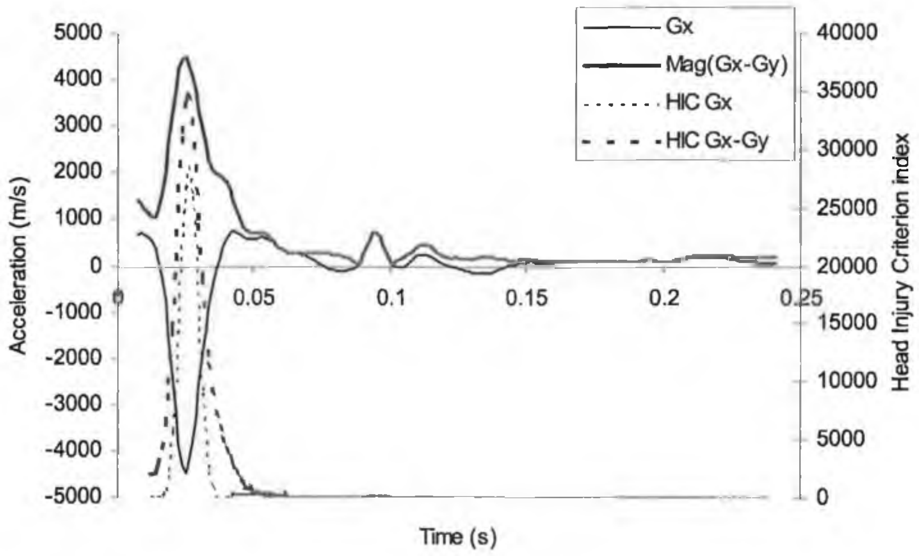


Figure C-6: Node 23 acceleration history of experiment wvb

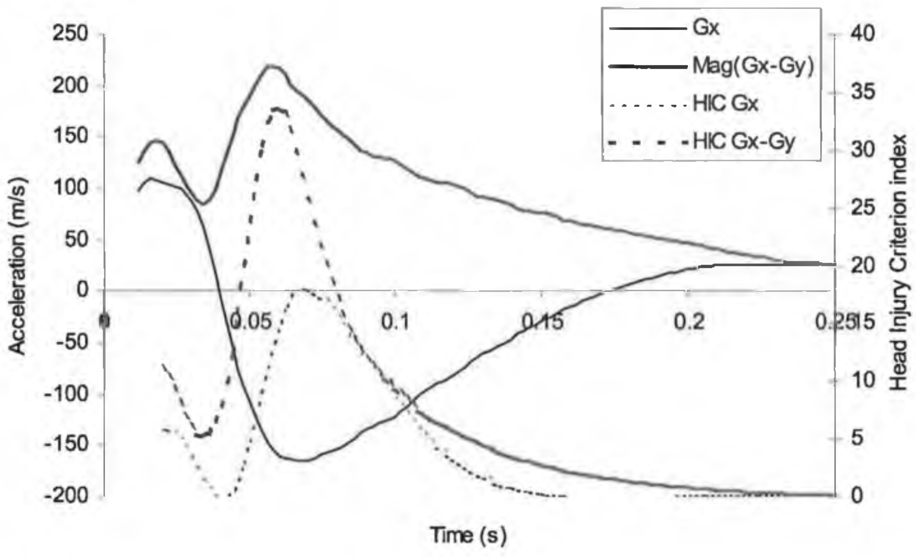


Figure C-7: Node 23 acceleration history of experiment wb

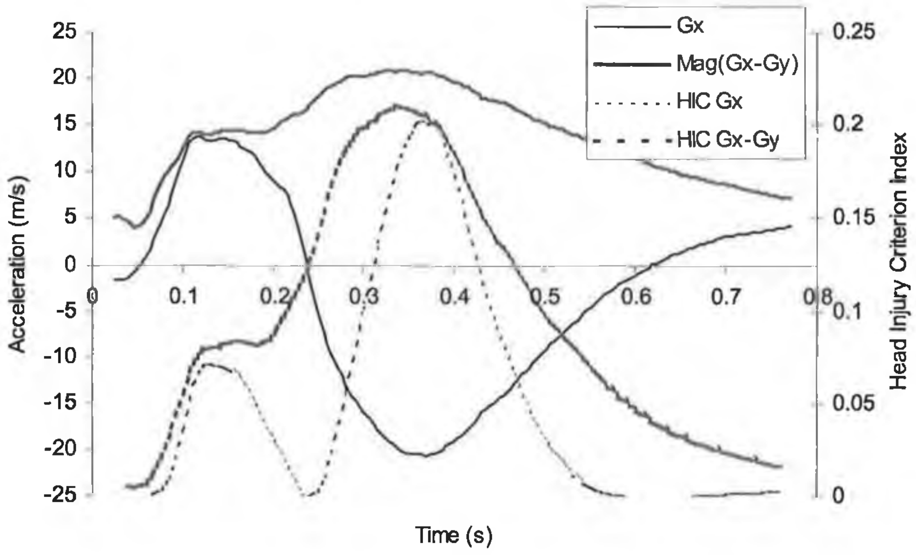


Figure C-8: Node 23 acceleration history of experiment w

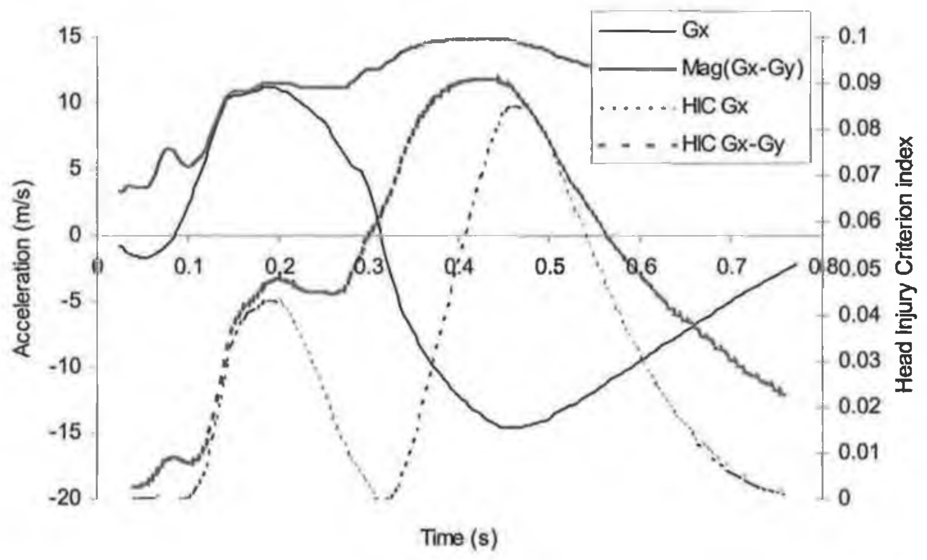


Figure C-9: Node 23 acceleration history of experiment h

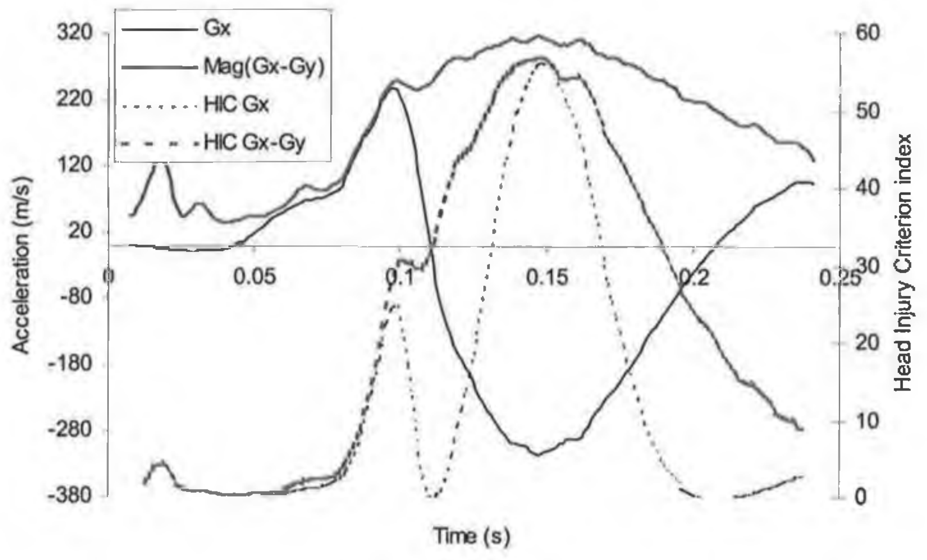


Figure C-10: Node 23 acceleration history of experiment hv

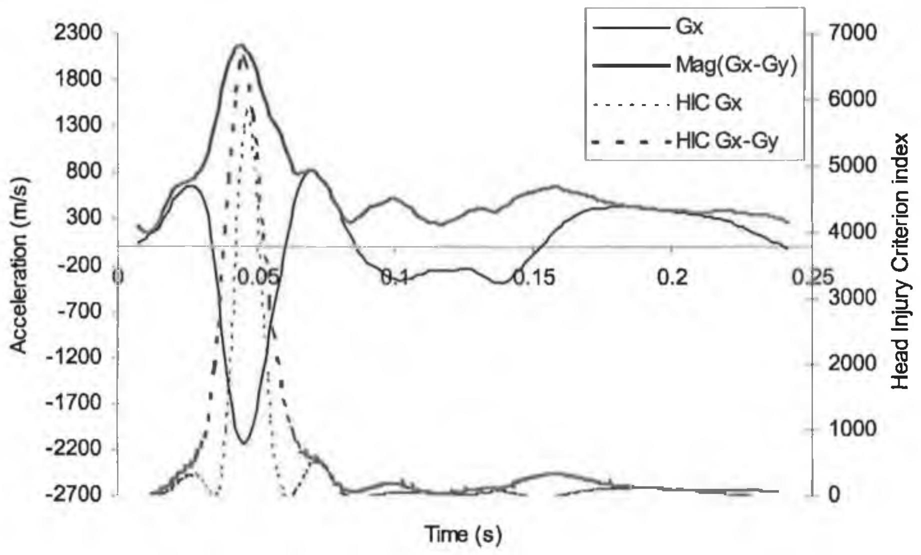


Figure C-11: Node 23 acceleration history of experiment hvb

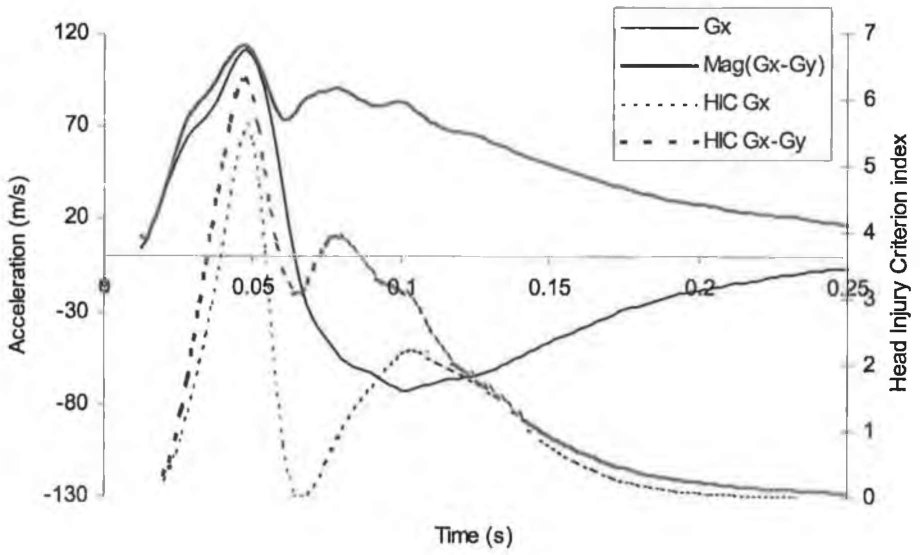


Figure C-12: Node 23 acceleration history of experiment hb

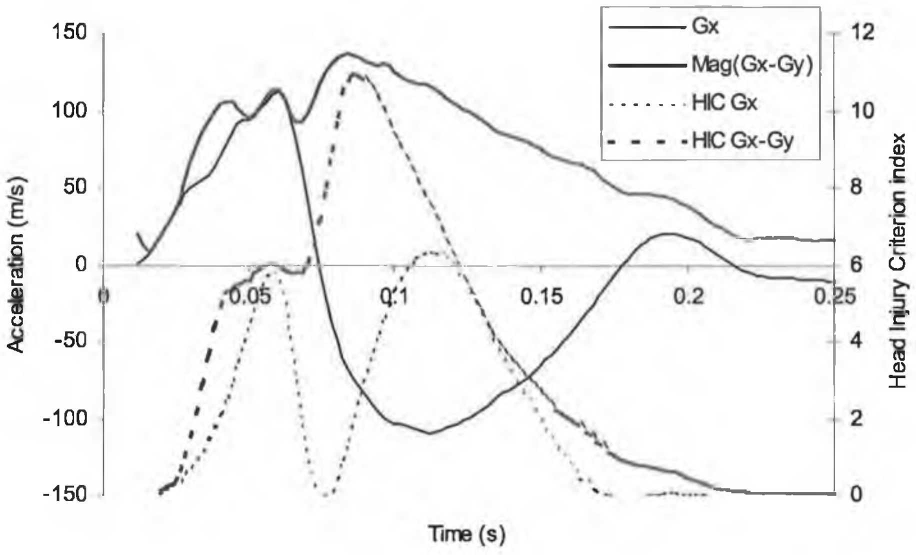


Figure C-13: Node 23 acceleration history of experiment hwb

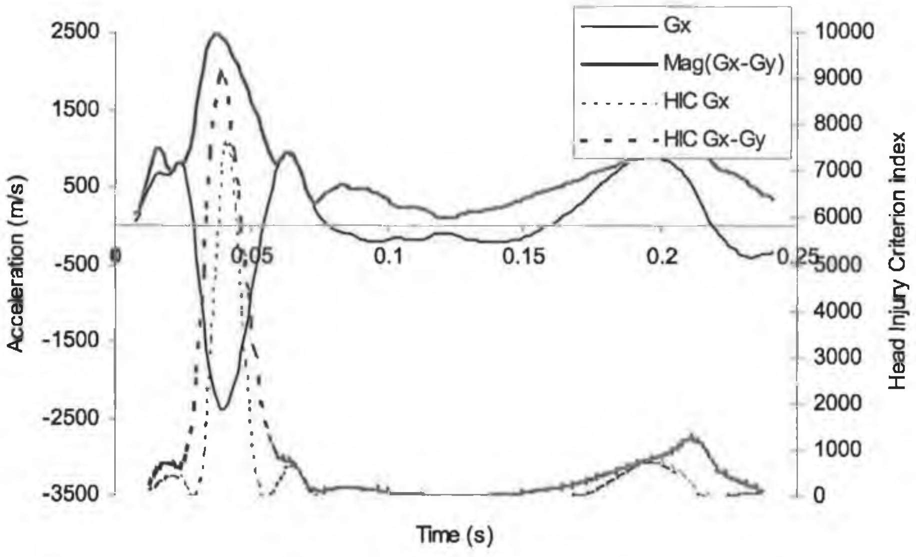


Figure C-14: Node 23 acceleration history of experiment hwvb

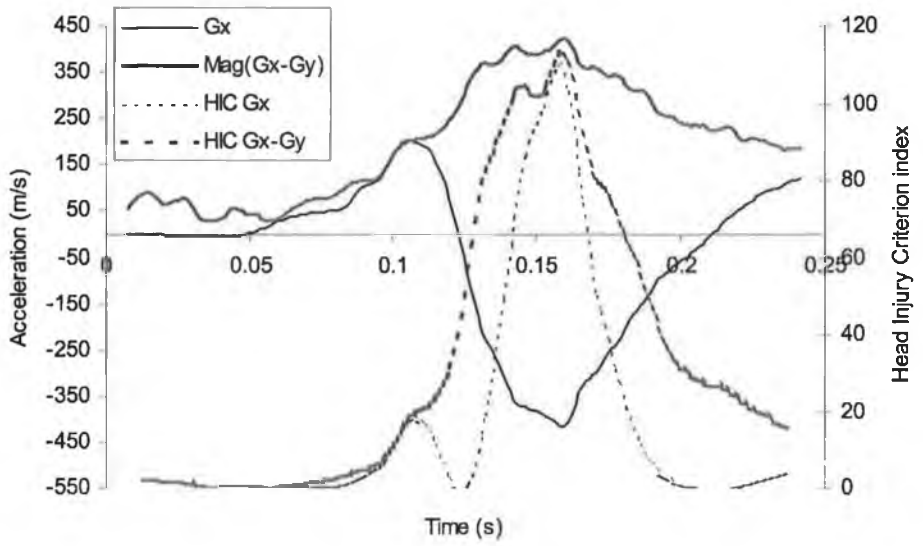


Figure C-15: Node 23 acceleration history of experiment hwv

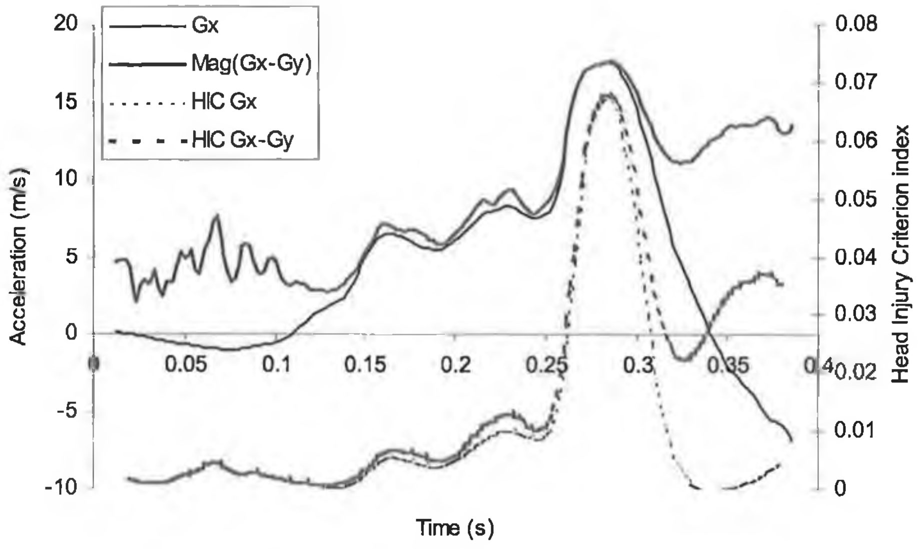


Figure C-16: Node 23 acceleration history of experiment hw

# Appendix D: MODERNAS KRISS 110

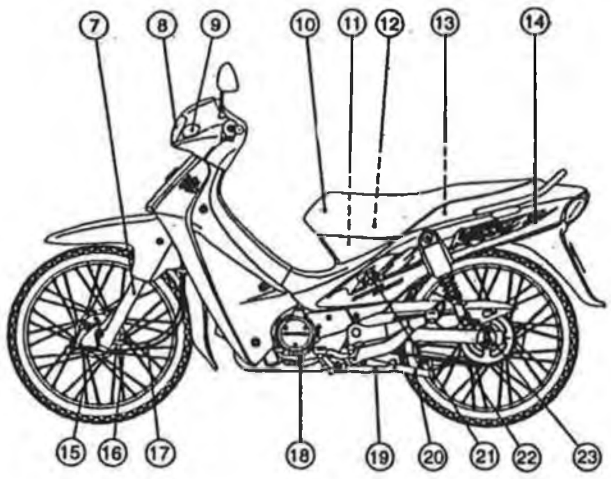


Figure D-1: KRISS 110 left hand view

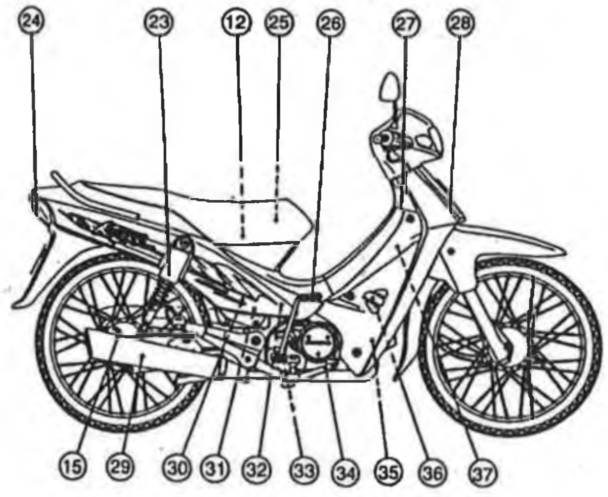


Figure D-2: KRISS 110 right hand view

Table D-1: Notations

| Number | Description |
|---|---|
| 7 | Front fork |
| 8 | Headlight |
| 9 | Turn signal light |
| 10 | Seat |
| 11 | Fuel tank |
| 12 | Helmet hook |
| 13 | Tool kit case |
| 14 | Seat lock |
| 15 | Brake lining wear |
| 16 | Brake cable |
| 17 | Speedometer |
| 18 | Shift pedal |
| 19 | Side stand |
| 20 | Center stand |
| 21 | Fuel tap |
| 22 | Drive chain |
| 23 | Rear shock absorber |
| 24 | Tail / brake light |
| 25 | Fuel tank cap |
| 26 | Kick pedal |
| 27 | Ignition switch / steering lock |
| 28 | Horn |
| 29 | Muffler |
| 30 | Fuse case |
| 31 | Battery |
| 32 | Rear brake light switch |
| 33 | Oil level gauge |
| 34 | Rear brake pedal |
| 35 | Spark plug |
| 36 | Carburetor |
| 37 | Air cleaner |

# Appendix E: TRL Crash Tests

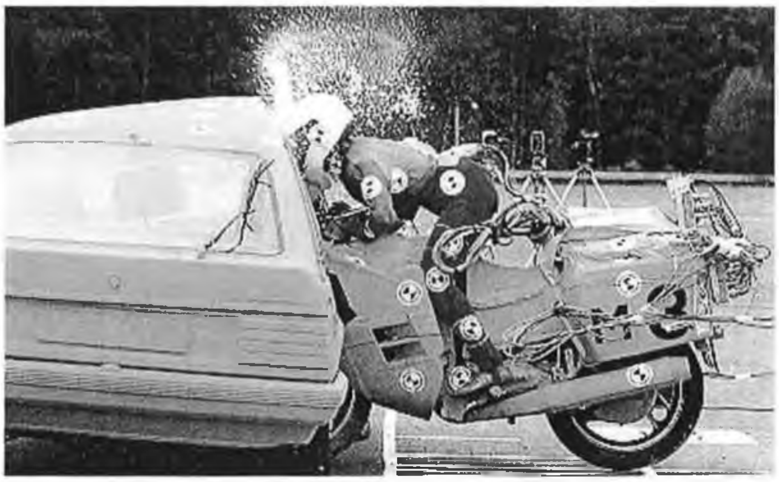


Figure E-1: One of the TRL full-scale tests

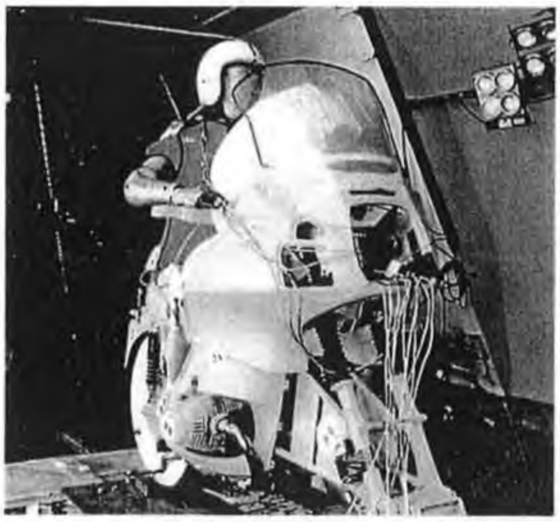
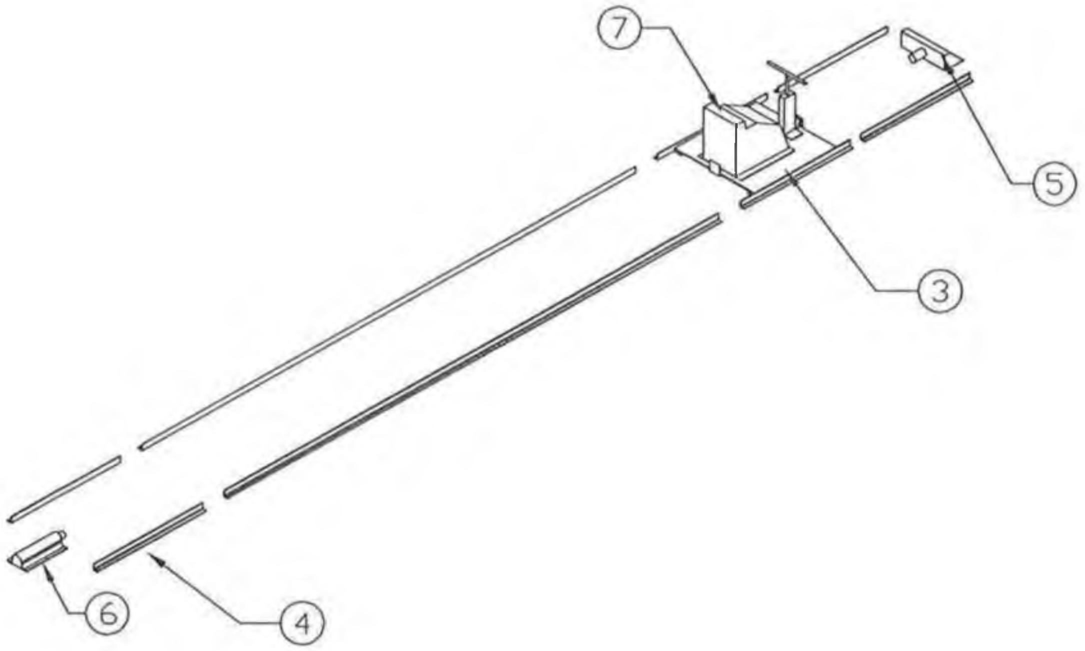
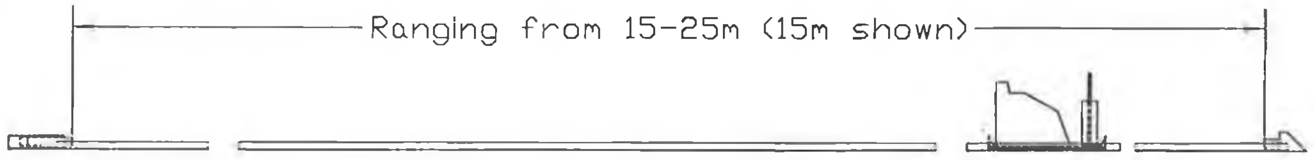


Figure E-2: Amended motorcycle in TRL sled test facility.

# Appendix F: Engineering Drawings of Sled Test Rig

| RevNo | Revision note | Date | Signature | Checked |
|---|---|---|---|---|
| | | | | |



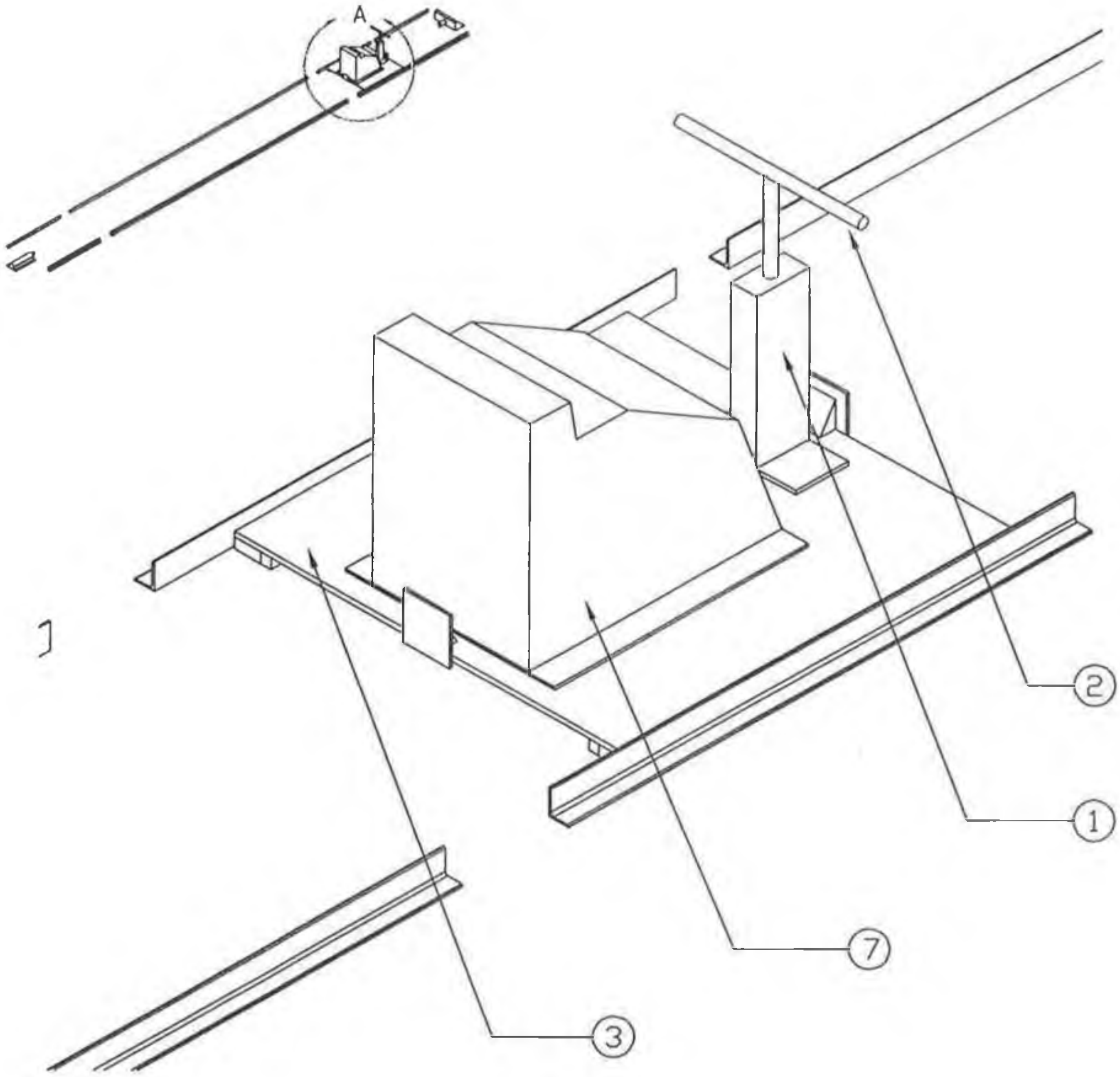


| ITEM | QTY | NAME | COMMENT |
|---|---|---|---|
| 9 | 1 | SPRING | |
| 8 | 1 | BLOCK | |
| 7 | 1 | FIXTURE | Initial fixture design for placing dummy |
| 6 | 1 | P_CYLINDER | pneumatic cylinder |
| 5 | 1 | SUB_IMPACT_BLK | impact block sub asm |
| 4 | 6 | RAIL | qty > 6, depends on design req |
| 3 | 1 | SUB_BASE | Moving Platform sub asm |
| 2 | 1 | FIXTURE3 | Simulate Obstacle |
| 1 | 1 | FIXTURE2 | Simulate Obstacle |

| Designed by | Checked by | Approved by - date | Filename | Date | Scale |
|---|---|---|---|---|---|
| Wong SV | | | | 19 Aug 1998 | 100:1 |

| | | | |
|---|---|---|---|
| Owner: Sch of Mech & Manuf Eng, Dublin City University, Dept of Mech & Manuf Eng, Universiti Putra Malaysia | Title/Name: Crash Rig Asm Dwg | | |
| | Drawing number | Edition | Sheet |

| RevNo | Revision note | Date | Signature | Checked |
|---|---|---|---|---|

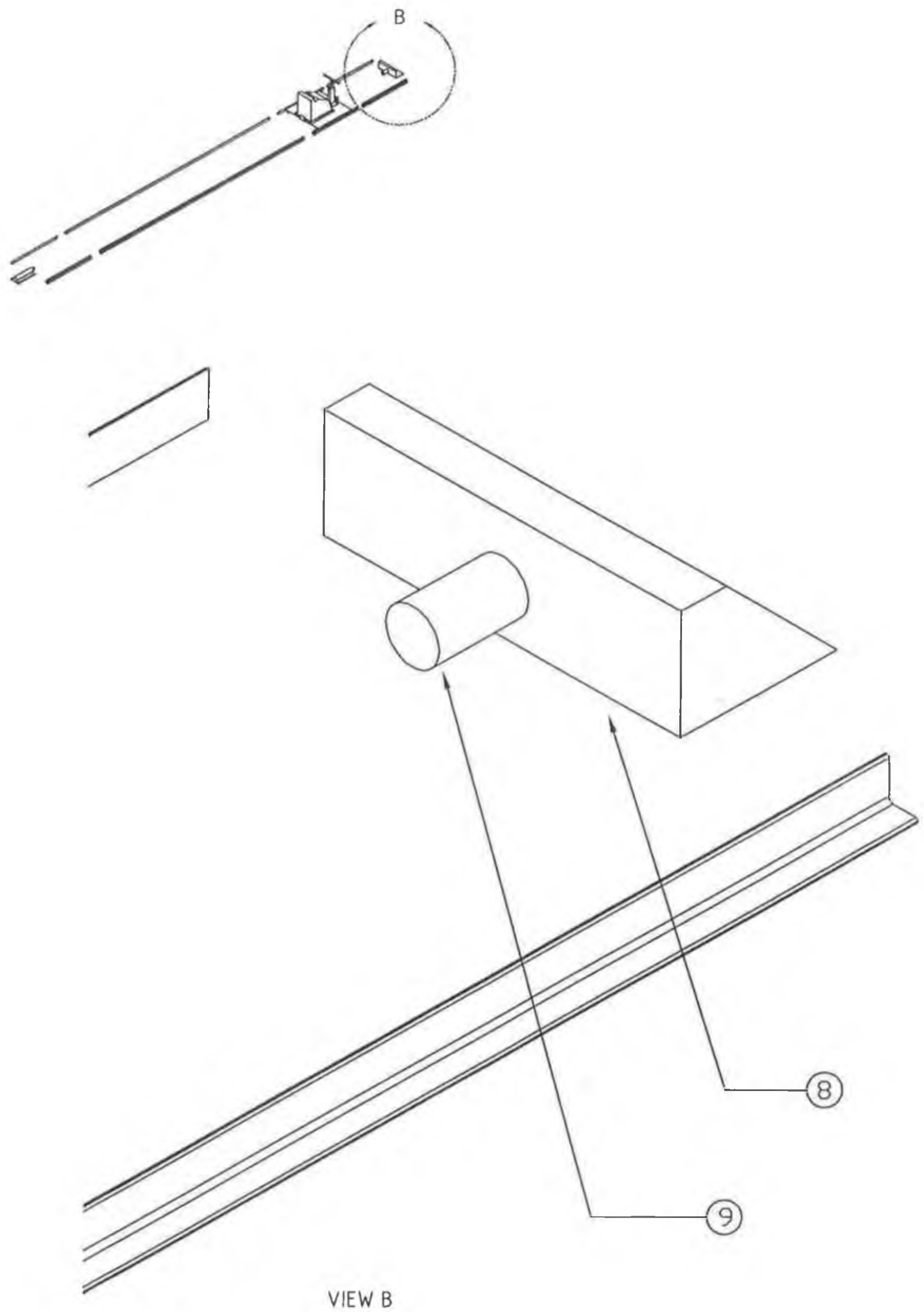


VIEW A

| Designed by | Checked by | Approved by - date | Filename | Date | Scale |
|---|---|---|---|---|---|
| Wong SV | | | | 19 Aug 1998 | 10:1 |

| Owner | Title/Name | | |
|---|---|---|---|
| Sch of Mech & Manuf Eng<br>Dublin City University<br>Dept of Mech & Manuf Eng<br>Universiti Putra Malaysia | Moving Platform & Fixtures (Iso view) | | |
| | Drawing number | Edition | Sheet |

| RevNo | Revision note | Date | Signature | Checked |
|---|---|---|---|---|
| | | | | |

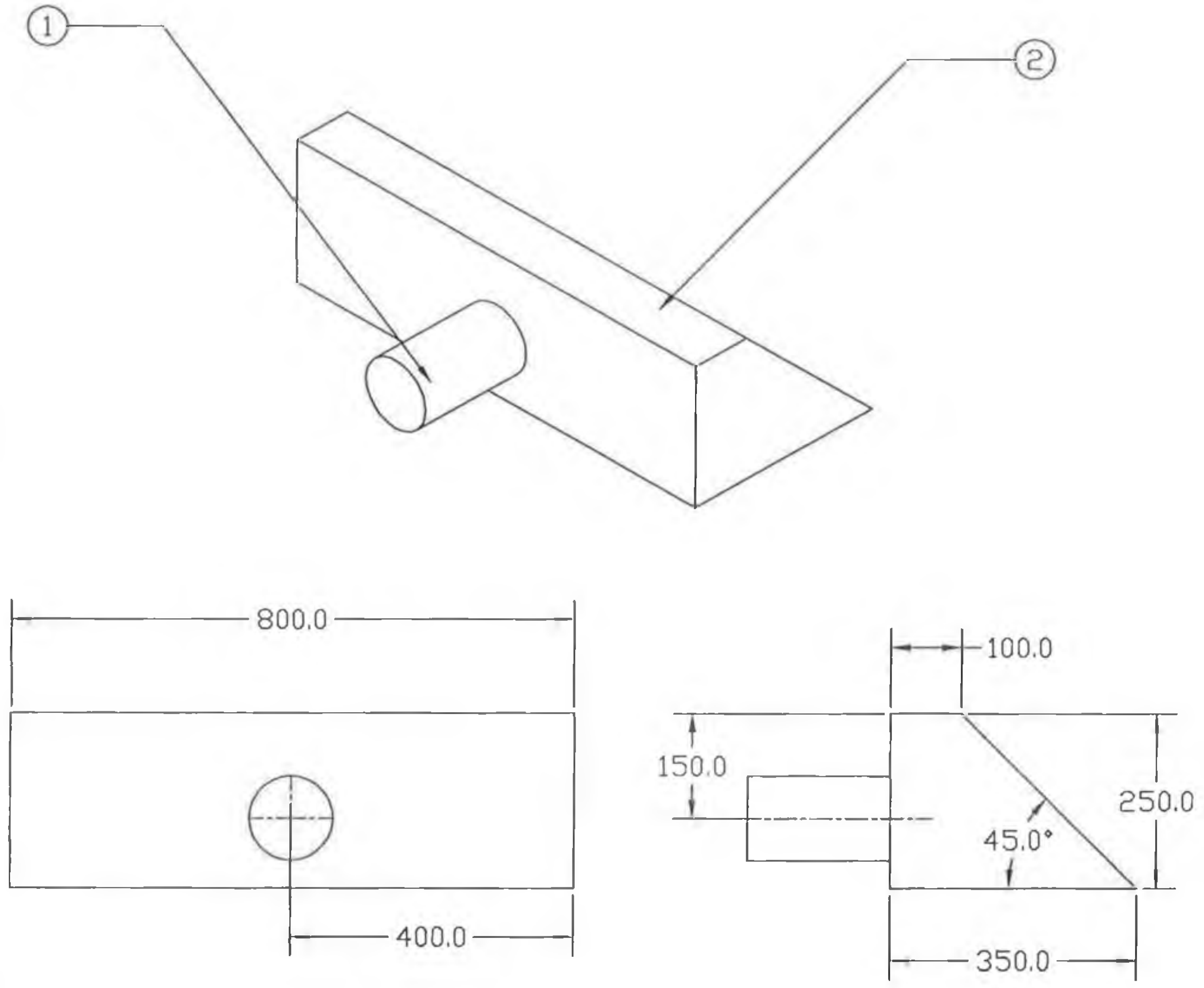


VIEW B

| Designed by | Checked by | Approved by - date | Filename | Date | Scale |
|---|---|---|---|---|---|
| Wong SV | | | | 19 Aug 1998 | 10:1 |

| | | | |
|---|---|---|---|
| Owner: Sch of Mech & Manuf Eng, Dublin City University, Dept of Mech & Manuf Eng, Universiti Putra Malaysia | Title/Name: Impact Block (Iso view) | | |
| | Drawing number | Edition | Sheet |

| RevNo | Revision note | Date | Signature | Checked |
|---|---|---|---|---|
| | | | | |

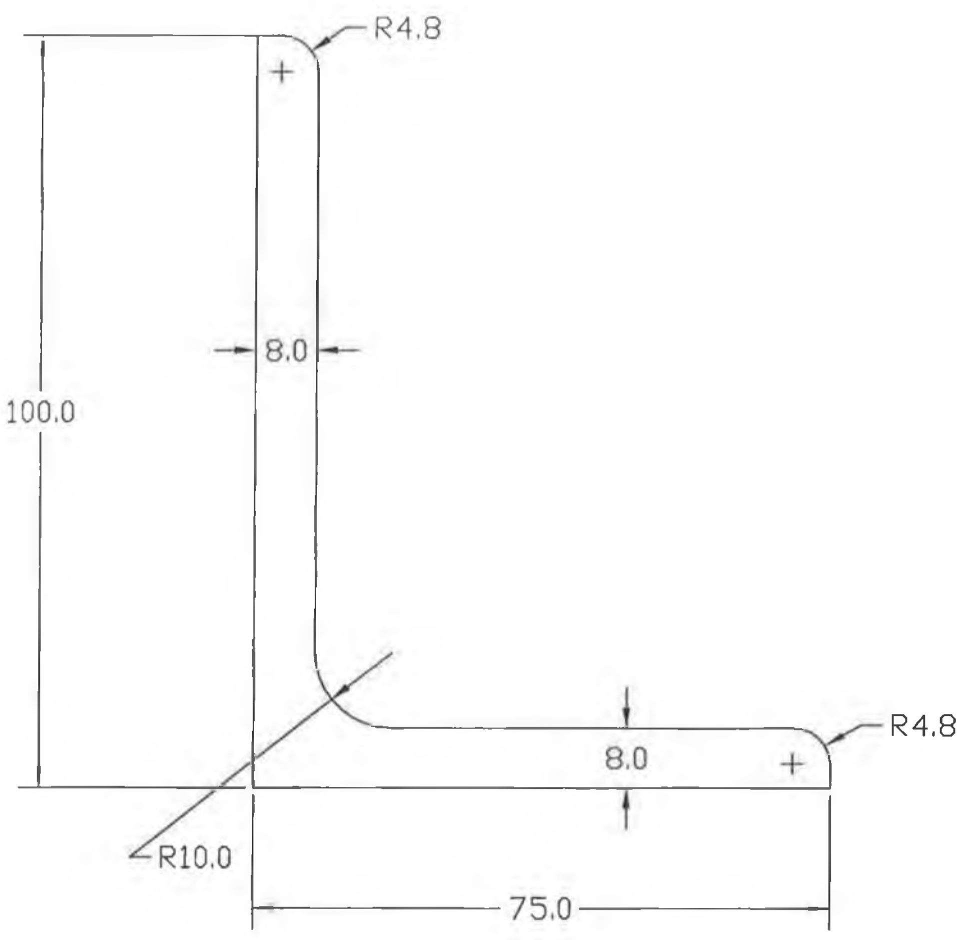


| ITEM | QTY | NAME |
|---|---|---|
| 2 | 1 | BLOCK |
| 1 | 1 | SPRING |

| | | | | | |
|---|---|---|---|---|---|
| Designed by Wong SV | Checked by | Approved by - date | Filename | Date 20 August 1998 | Scale 10:1 |

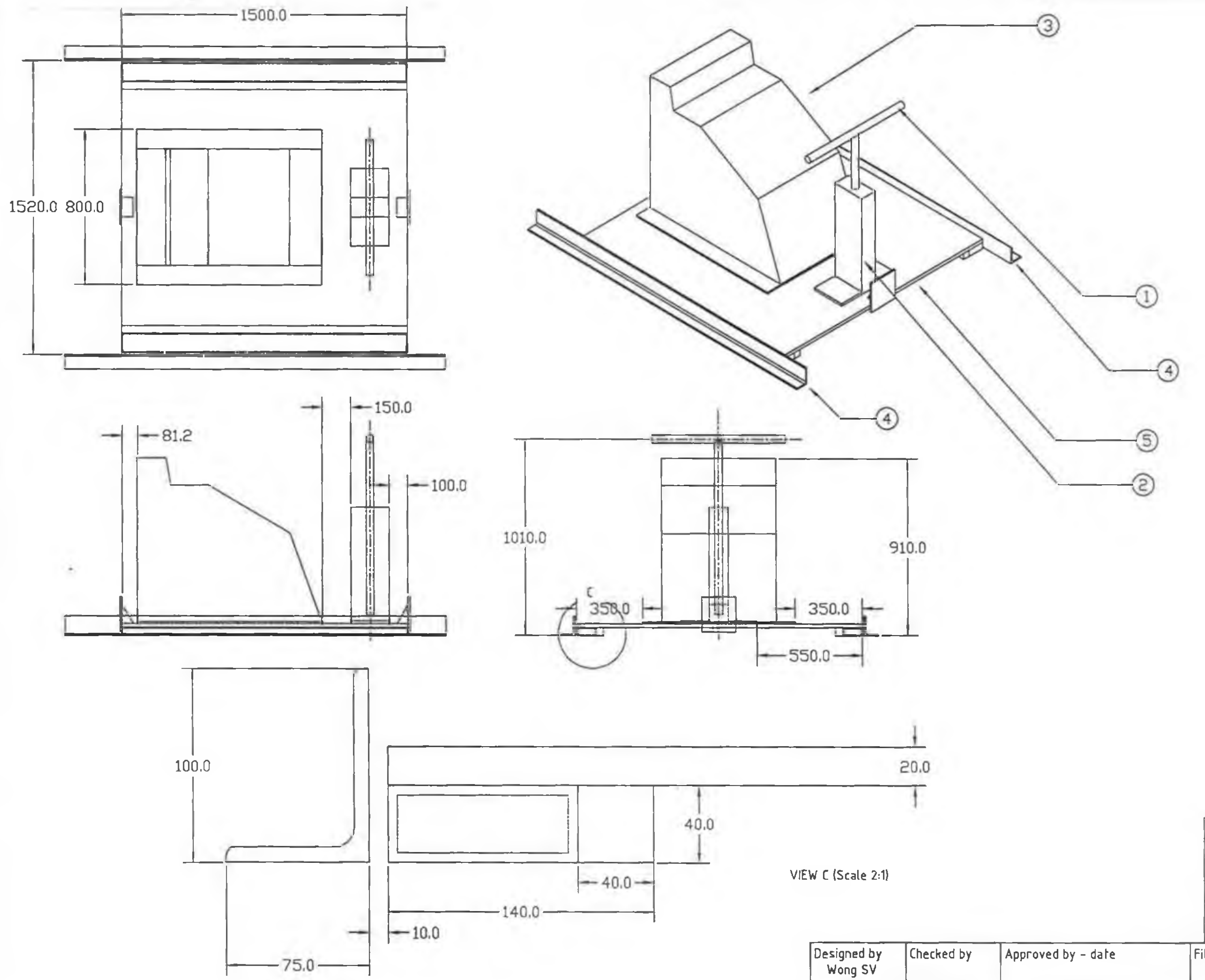
| | | | |
|---|---|---|---|
| Owner: Sch of Mech & Manuf Eng Dublin City University Dept of Mech & Manuf Eng Universiti Putra Malaysia | Title/Name Impact Block asm dwg | | |
| | Drawing number | Edition | Sheet |

| RevNo | Revision note | Date | Signature | Checked |
|---|---|---|---|---|
| | | | | |



| Designed by | Checked by | Approved by - date | Filename | Date | Scale |
|---|---|---|---|---|---|
| Wong SV | | | | 19 Aug 1998 | 1:1 |

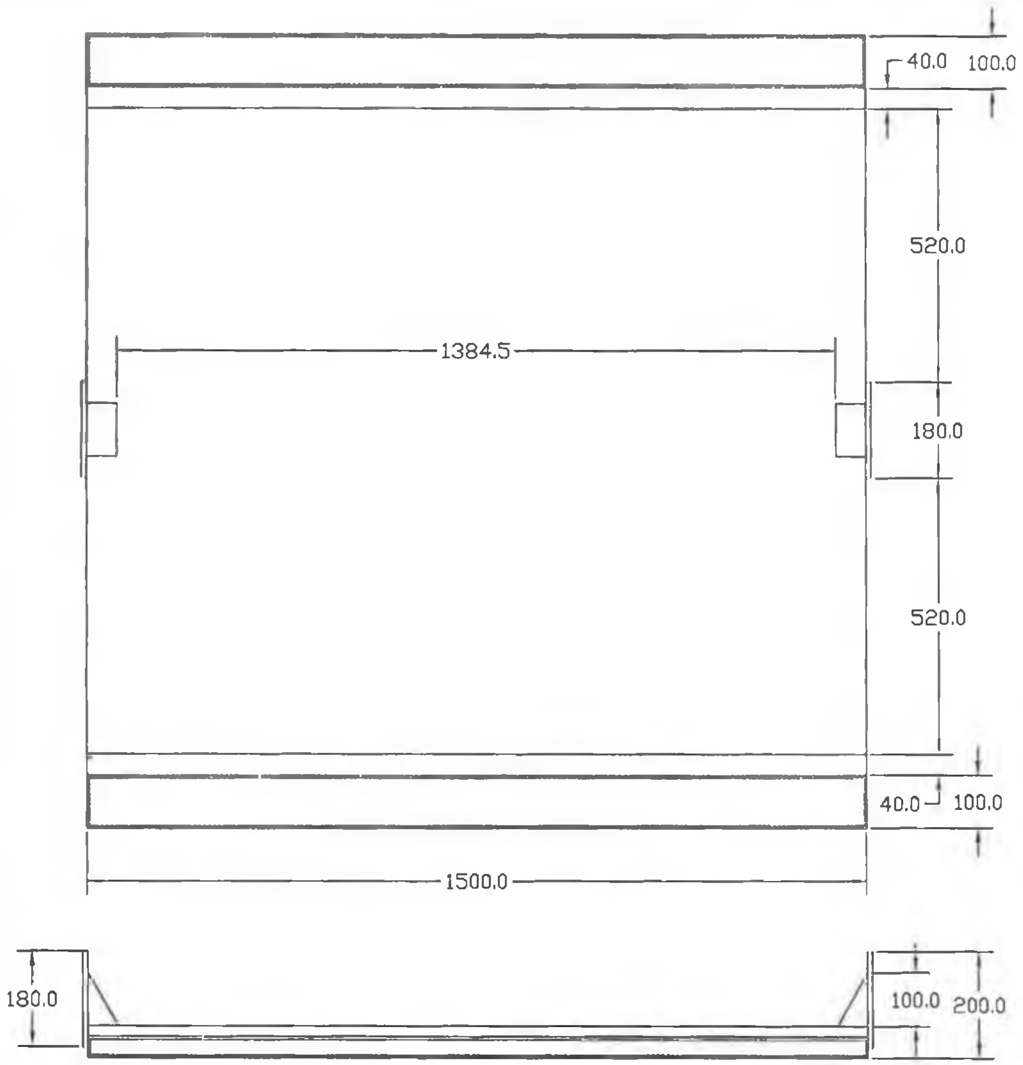
| | | | |
|---|---|---|---|
| Owner: Sch of Mech & Manuf Eng, Dublin City University, Dept of Mech & Manuf Eng, Universiti Putra Malaysia | Title/Name: Rail front view (BS 4848: Part4: 1972, 100x75x8) | | |
| | Drawing number | Edition | Sheet |



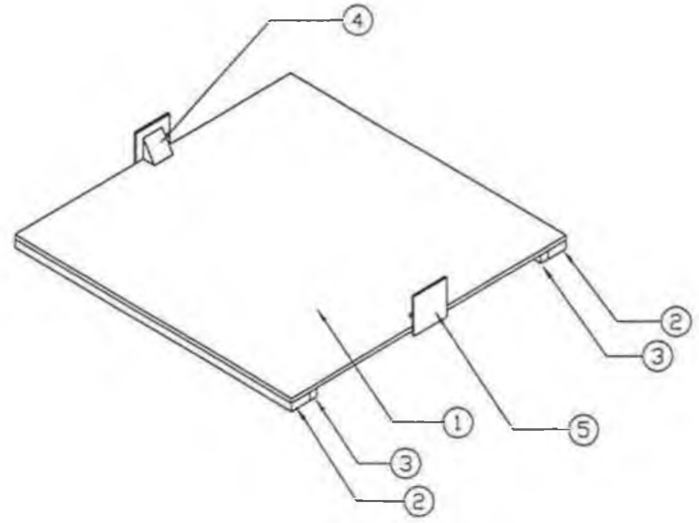
| | | |
|---|---|---|
| 5 | 1 | SUB_BASE |
| 4 | 6 | RAIL |
| 3 | 1 | FIXTURE |
| 2 | 1 | FIXTURE2 |
| 1 | 1 | FIXTURE3 |
| ITEM | QTY | NAME |

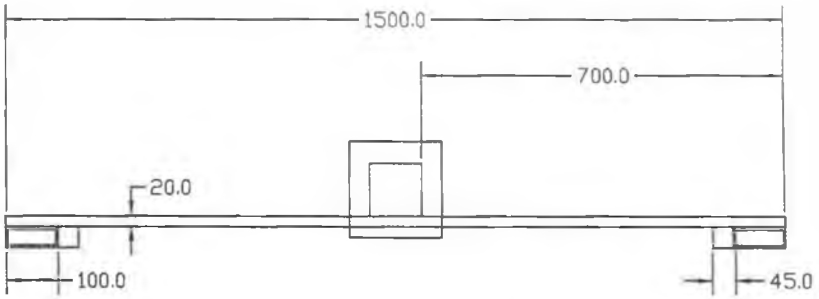
| Designed by Wong SV | Checked by | Approved by - date | Filename | Date 19 August 1998 | Scale NTS |
|---|---|---|---|---|---|
| Owner Sch of Mech & Manuf Eng Dublin City University Dept of Mech & Manuf Eng Universiti Putra Malaysia | Title/Name Moving Platform, Fixtures and Rails asm dwg | | | | |
| | Drawing number | | | Edition | Sheet |

| RevNo | Revision note | Date | Signature | Checked |
|---|---|---|---|---|



|RevNo|Revision note|Date|Signature|Checked|
|---|---|---|---|---|





| | | | | | |
|---|---|---|---|---|---|
| | | | 4 | 2 | IMPACT_PLTSUP |
| | | | 3 | 2 | BASE_SUP |
| | | | 2 | 2 | AIR_BEARING |
| 5 | 2 | IMPACT_PLT | 1 | 1 | BASE |
| ITEM | QTY | NAME | ITEM | QTY | NAME |

| | | | | | |
|---|---|---|---|---|---|
| Designed by Wong SV | Checked by | Approved by - date | Filename | Date 20 August 1998 | Scale NTS |
| Owner Sch of Mech & Manuf Eng Dublin City University Dept of Mech & Manuf Eng Universiti Putra Malaysia | Title/Name Moving Platform asm dwg | | | | |
| | Drawing number | | | Edition | Sheet |