CAD Modeling, Multibody System Formalisms and Visualization – An Integrated Approach

A. Daberkow, E. Kreuzer, G. Leister and W. Schiehlen
Institute B of Mechanics, University of Stuttgart

1 Introduction

The increasing use of software in multibody dynamics and its application to engineering design and analysis requires an efficient management of the communication between software tools. As product life time is shrinking, shorter periods for design require an automated model data exchange and simulation process for a dynamic analysis.

In this paper an integrated approach of CAD-(Computer Aided Design) modeling, generation of equations of motion, simulation and visualization of multibody systems is described. An object-oriented data model for different multibody formalisms is integrated in a commercially available CAD-3D-system. With respect to existing CAD-interfaces, different solid model design methods and various visualization demands the datamodel allows multibody modeling with a direct interface to a data base. Different software tools like an integrated Newton-Euler formalism are able to use immediately the parametrized multibody system data base. For multibody systems with closed kinematic loops a set of ordinary differential equations and decoupled algebraic equations is formulated automatically which can be solved with explicit multistep integration algorithms. This is achieved by a minimal set of generalized coordinates being specified during the numerical integration. A additional interface provides data for visualization from the simulation tool.

The basic steps and the extreme flexibility of this automated mechanical design and simulation process is demonstrated for a crank-slider mechanism.

W. Schiehlen (ed.), Advanced Multibody System Dynamics, 87–106.

2 Object-oriented data model for multibody systems

Modeling of a mechanical system by the method of multibody systems is characterized by a composition of rigid bodies, joints, springs, dampers, and servomotors, see Figure 1. Force elements like springs, dampers, and servomotors acting in discrete nodal points result in applied forces and torques on the rigid bodies. Joints with different properties connecting the various bodies constrain their motion, they are often identified as constraint elements.

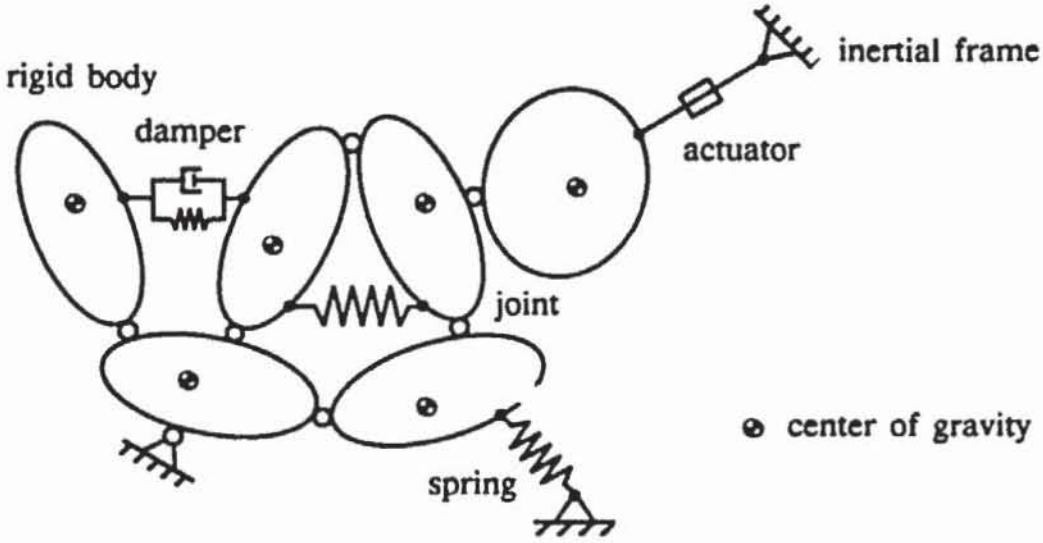

Figure 1: Multibody System

For the generation of the equations of motion computer programs may be used. Well known multibody system computer codes producing exclusively numerical data are ADAMS, Orlandea [16], and DADS, Haug [4]. To the contrary, computer programs like SD-FAST, Rosenthal and Sherman [29] and NEWEUL, Kreuzer [8] provide the explicit symbolical expressions for the system equations. A survey of the different formalisms and computer codes in multibody dynamics can be found in Schiehlen [25] and in Roberson and Schwertassek [22].

Nowadays CAD-systems are widely embedded in the industrial design and construction process, while a general application of three-dimensional CAD-systems is still rare. They support an analytically and topologically complete modeling, a collision detection, and the calculation of surface and volume properties closely related to the geometric representation of solid models, see Mortenson [15] and Pahl [20].

Some couplings of solid modelers with multibody simulation software are realized for the numerical computer code ADAMS, e.g. for the CAD-system ARIES [2]. A CAD-3D-system independent approach is included in the program package RASNA and is described by Hollar and Rosenthal [6].

A system dynamics analysis requires as basic parameters mass, center of gravity, and moments of inertia of each body related to the geometry model and modeling method of the CAD-system used. A modular software concept demands an exchange of complete or single object data between the CAD-system and the multibody formalism. Therefore, a general interface to multibody computer codes is demanded to serve as a compatible and comfortable CAD-post processor, taking the different algorithms and implementations of multibody computer codes into account. The commercially available multibody modeling software tools within CAD-systems are mostly dedicated to a particular multibody dynamics computer code. Often, no options are supplied for a parametric multibody system description or the modeling is restricted to either robot, mechanism or vehicle dynamics. This variety of systems, each with different model data and the growing problems in the exchange of data, requires the development and production of cheaper and more reliable software products.

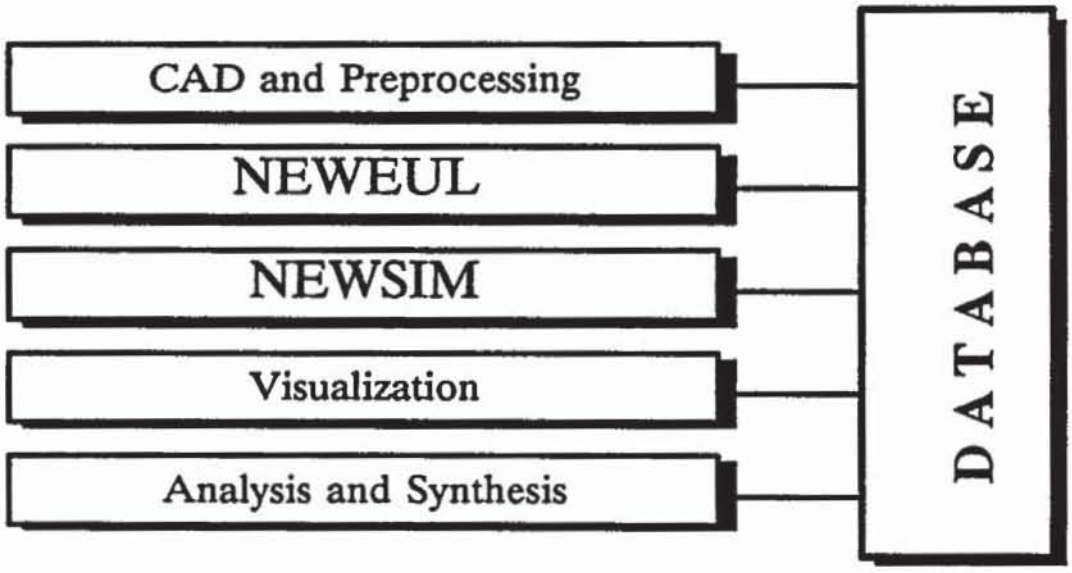

Figure 2: Modules within the database concept

Consequently this leads to a database concept for the CAD-3D-modeling of multibody systems, see Figure 2:

- Collect the necessary data describing uniquely a multibody model for the different multibody programs.
- Examine the different geometry models of CAD-systems for solids and extract the relevant data for multibody systems.
- Define a geometry model for the representation of multibody elements.
- Design data types and operations and construct a software interface for a code-independent modeling of multibody systems.

A dynamic simulation environment for multibody systems represents in practice a large, sophisticated software system. Therefore, an important step is the definition of an abstract data model on a conceptual level. A first effort to develop a generalized data model for multibody systems including symbolical parameters and a postprocessing of CAD-data is described by Otter, Hocke, Daberkow, and Leister [17]. Each of the bodies is described by body-fixed reference frames. Further body-fixed frames, related joints and force elements are described. Additional symbolical parameters are defined for the position and orientation of the frames with respect to each other as well as the mass properties of the bodies. Consequently, for symbolical as well as numerical formalisms a generalized data base relies upon the basic modeling elements frame, body, joint, and force and is further adapted and extended with respect to the geometry models in CAD-3D and graphics systems, see Daberkow [3].

The boolean combination of two or more primitive objects to a new solid object is the main characteristic of Constructive Solid Geometry (CSG), Figure 3. For two-dimensional projections of the CSG model, an equivalent wire or face model has to be derived from the binary tree of the primitives and their transformations.

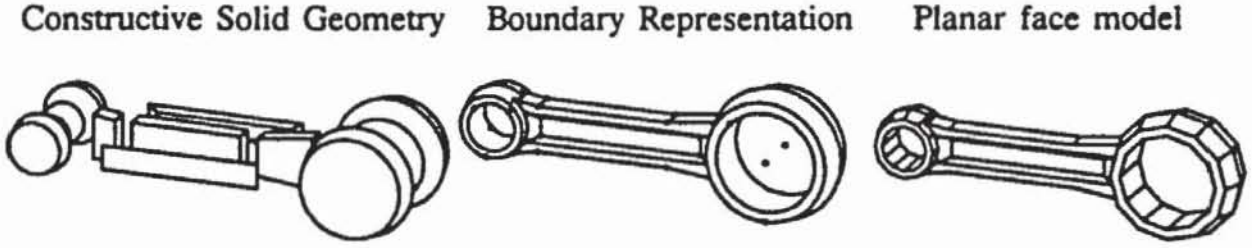

Figure 3: Geometric modeling approaches

A Boundary Representation (B-Rep) model, see Figure 3, allows the boolean combination of primitive objects, too. Each primitive object and the actual modeling state is described by a complete spatial boundary, whose topological validity may be checked by application of the Euler operators to the enclosing faces, edges, and vertices, see Mortenson [15]. The solid modeling tool PARASOLID uses a boundary representation and is commercially available in many CAD-3D-systems.

A simple planar face model, see Figure 3, as a special case of the B-Rep, serves as a geometry model which is suitable for high-speed 3D-visualization, see Schiehlen and Daberkow [26]. Moreover, this model is implemented in graphic standards like PHIGS, and graphic languages like Iris GL.

A property of a solid can be derived from a face normal specifying the inner and outer parts of an object, while the coincidence of the vertices of adjoining faces is not guaranteed. The geometric modeling by parametrized shapes is appropriate for geometric objects, whose shape is uniquely defined by a restriced number of

parameters. Examples of parametrized shapes with an equivalent wire representation are shown in Figure 4.

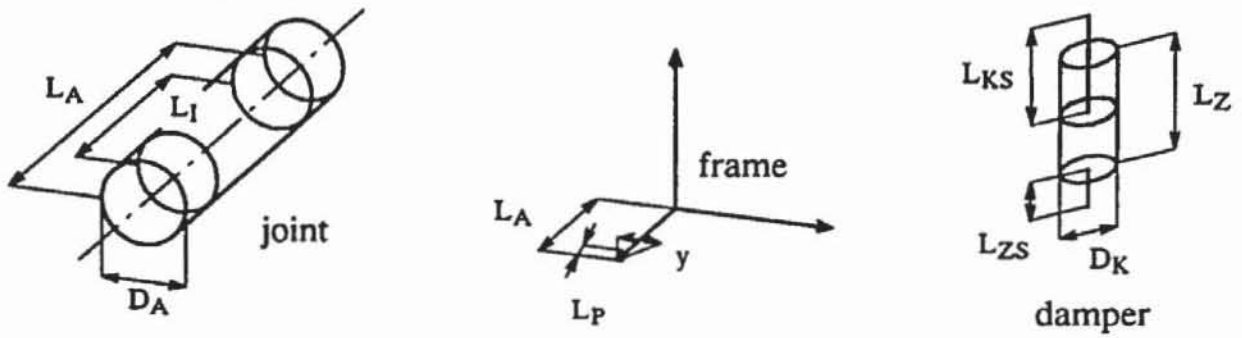

Figure 4: Parametrized wire representations of multibody elements

For the global properties volume, surface area, moment of inertia, and center of gravity of solid models, integrals have to be evaluated like

$$I = \int_{Solid} f^V dV \tag{1}$$

see e.g. Mortenson [15], where $f^V = f^V(x, y, z)$ denotes a scalar property function. While Constructive Solid Geometry suggests the calculation of mass properties by the following recursively applied formulas

$$\begin{aligned} \int_{Solid1 \cup Solid2} f^V dV &= \int_{Solid1} f^V dV + \int_{Solid2} f^V dV - \int_{Solid1 \cap Solid2} f^V dV, \\ \int_{Solid1 - Solid2} f^V dV &= \int_{Solid1} f^V dV - \int_{Solid1 \cap Solid2} f^V dV, \end{aligned} \tag{2}$$

boundary representations allow the evaluation via surface integrals.

The examination of different geometry models yield the following results:

- Mass property calculation modules for multibody systems do not depend on the model geometry (CSG or B-Rep). These results can be related directly with the input entities needed for the rigid bodies.
- A planar face model derived from the geometric entities of the solid body yield the graphic data for the description of the body's shape necessary for visualization.
- The parametrized shapes are well suited to serve as a geometry model for multibody modeling elements like frame, joint, and force.

The object-oriented data model conceptually developed by Otter et al. [17] results in classes defined for the elements *frame*, *body*, *joint*, *force*, *interact*, *global*, and *param* and additional operations valid for these classes.

An object of class *part* e.g. Figure 5 comprises alle time-invariant data of a rigid body. It is obvious that the components inertia matrix and mass of an object of class body are supplied by their numerical values. A location of the center of gravity different from the body-fixed reference frame is taken into consideration by reference to an equivalent object of class *frame*.

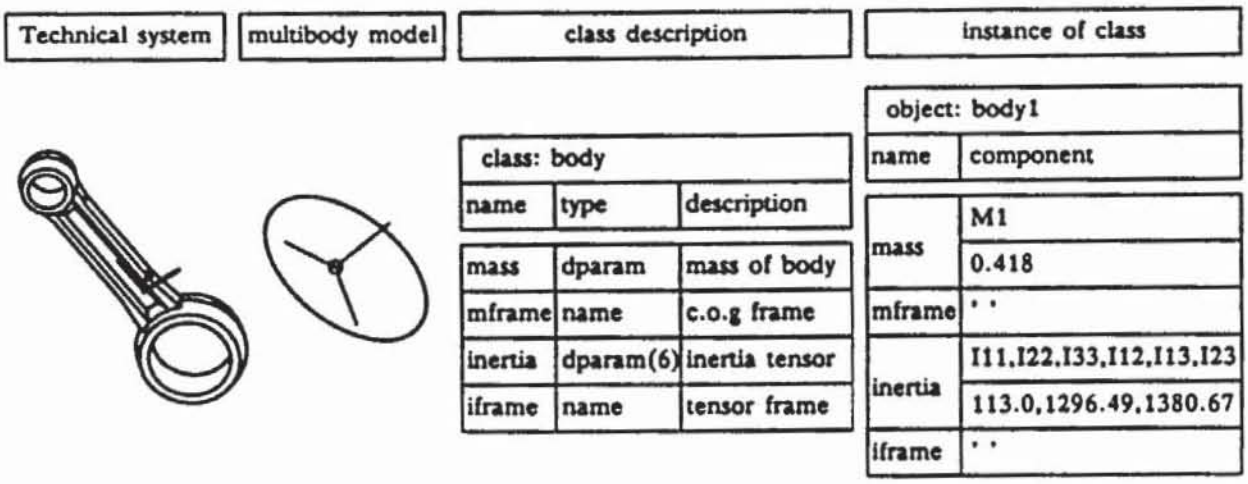

Figure 5: Object of class *body* with its data model

Coupling elements of a multibody system are collected in class *interact*. Interactions are valid between two objects of class *frame* on different objects of class *part*.

Due to object-oriented software techniques, the definition of abstract data types in classes furthermore demands a description of the operations valid on the objects, see e.g. Meyer [14]. These operations are designed for a practical, interactive multibody modeling process, e.g. in a CAD-3D-system. For all classes the basic operations 'create', 'delete', 'modify', and 'list' are defined, more complex operations take the relationships between objects of a multibody system into account.

Further classes are required for the graphical representation, like the actual frame axis length, its color or visibility, which depend on the actual multibody size and modeling state. An equivalent geometry data model for multibody elements well suitable for machine, robot and vehicle dynamics requires a unique spatial representation of the multibody elements, their function and physical quantity, see Daberkow [3]. From Figure 4 it is obvious that spatial parametrized shapes satisfy a graphic representation for objects of class *frame*, *joint*, and *force*. The definition of the geometry 3D classes *g3frame*, *g3joint* and *g3force* and operations for the geometry data model is equivalent to the multibody data model and includes classes comprising color, projection and viewpoint data.

3 Implementation and CAD-3D-realization

The implementation of the object-oriented data model in the data base system RSYST [23] allows storage and modification of multibody system objects. To realize fast access and interactive graphic visualization, the implementation of the object-oriented classes and operations within the CAD-3D-system is performed by means of data types and routines, which result in a system-independent modeling kernel library for multibody systems, see Daberkow [3]. This high level library DAMOS-C (DAta MOdel Standard implemented in C) supplies interfaces for modeling, input, and output as well as for the graphic representation. This open interface allows the integration in the commercially available CAD-3D- system SIGRAPH [28] and a new developed graphics-system.

The integration scheme in Figure 6 shows the interfaces to the CAD-3D software moduls of SIGRAPH. An extension of the CAD command language supplies additional commands which are necessary for the execution of multibody modeling operations. The CAD-3D-system menu is completed by special multibody system icons. To assure the graphic display of the modeling elements, the parametrized shapes are modeled via the 3D-wireframe entities of the CAD-graphic subsystem. A multibody command language of RSYST serves as a multibody system neutral file to store the multibody objects, see Otter et al. [19]

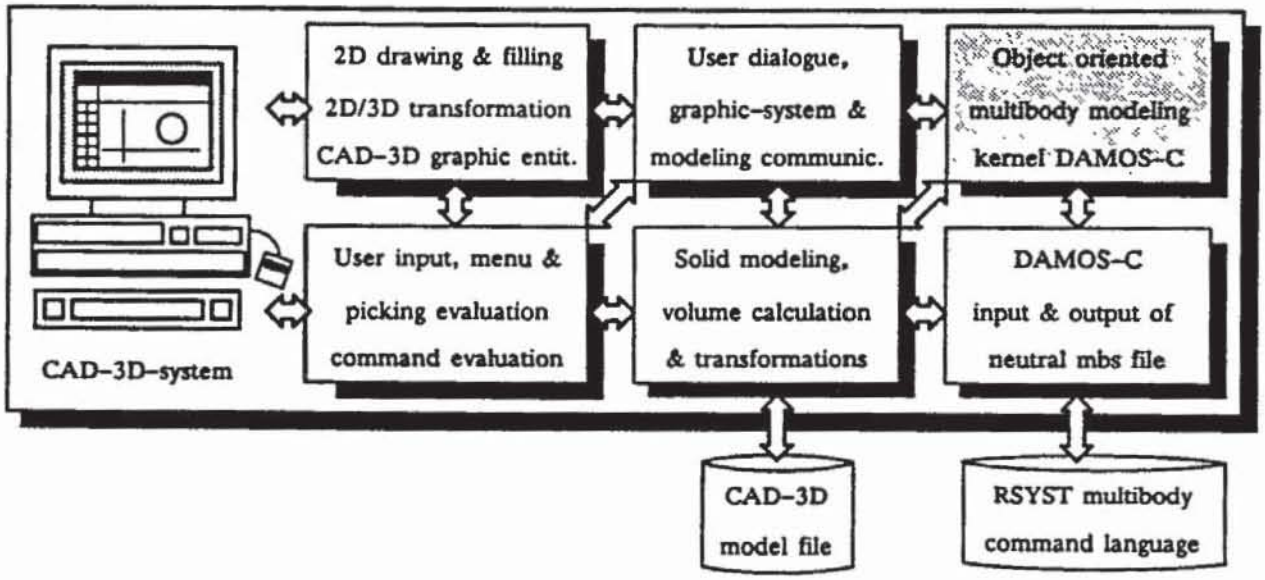

Figure 6: Integration of the multibody modeling kernel

The solid model design of a crank slider mechanism is performed by volume oriented techniques in PARASOLID from a disassembled model, Figure 7. All bodies of the

crank-slider mechanism of a single four stroke engine are shown in Figure 7. Each body is supplied with adequate density attributes.

The first multibody modeling step is the initialization. Here, an appropriate solid is chosen as the inertial body of the multibody system, see Figure 7. In the next step arbitrary solids are interactively chosen to have the properties of a multibody part. Each object of class *part* retrieves its mass and inertia components from the mass property calculation modul of PARASOLID. To visualize the multibody part property, the equivalent solids are supplied by reference frames, located in the center of gravity.

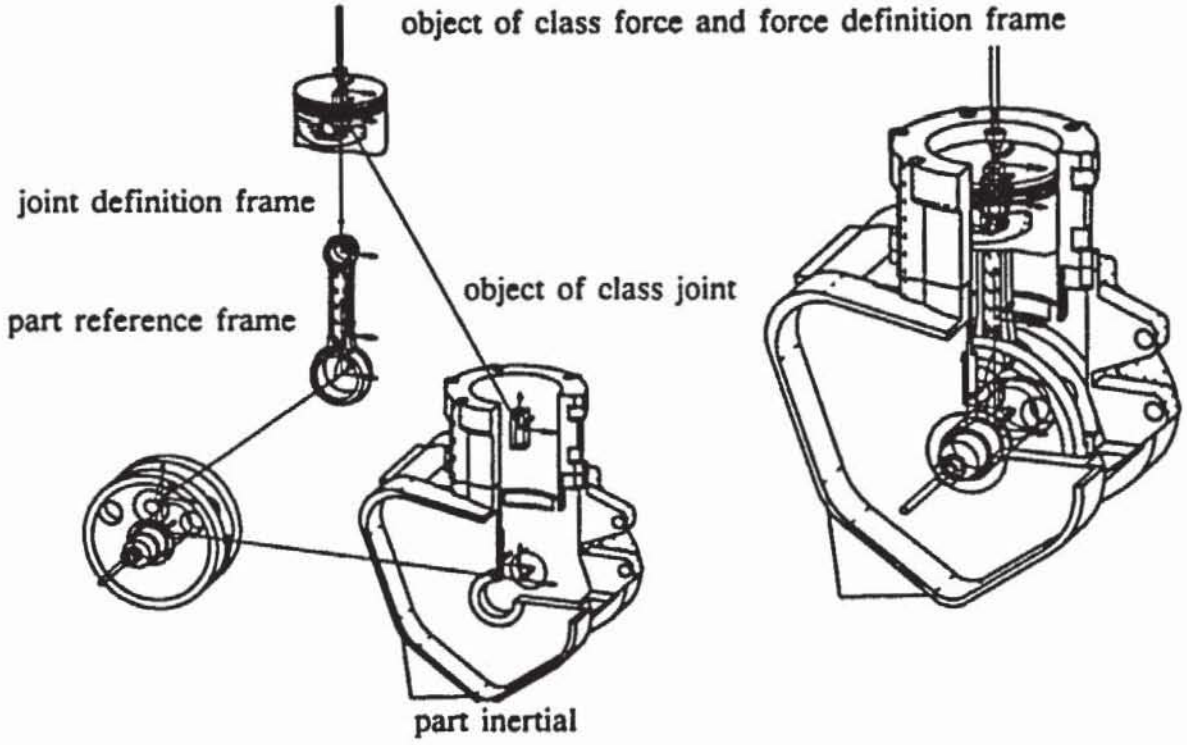

Figure 7: Disassembled and assembled mechanism with joint and force objects

By default, the orientation of further created joint and force definition frames is parallel to the specified reference frame. The position of these frames is defined by the CAD-3D-picking commands performed by the user. Figure 7 shows these modeling steps and the graphic representation of the objects. Joint definition frames are located along the unit normals of those faces, which form bearing surfaces or bearing bores of a solid.

A planar system modeled for spatial analysis demands a proper constraint selection. Redundant constraints remain if a mechanism is supplied with joints of class *revolute* and *translational*, making the determination of reaction forces impossible. Consequently, for an analysis modified joints have to be chosen. Objects of class *revolute* are visualized by the parametrized shapes and the wireframe entities. The connec-

tion between the objects of class *part* by the object of class *interact* is visualized by a 3D-line entity between the interacted frames.

The multibody modeling kernel library implemented in the CAD-3D-system supports an assembling of arbitrary pairs of class *part*. Figure 7 shows the assembling of individual solids over the equivalent objects of class joint. By modifying the **rangle** component of arbitrary objects of class *joint*, an initial multibody configuration is adjusted interactively, providing therefore an initial estimate for closed loop systems. Finally, an object of class *force general* is added to the piston part.

The multibody model conversion from the extended CAD-3D-database to a multibody computer code is realized by the neutral file [19]. An integrated RSYST multibody modul MBSDIA, see Hocke [5], generates the multibody model data base for further analysis and simulation, see Figure 8.

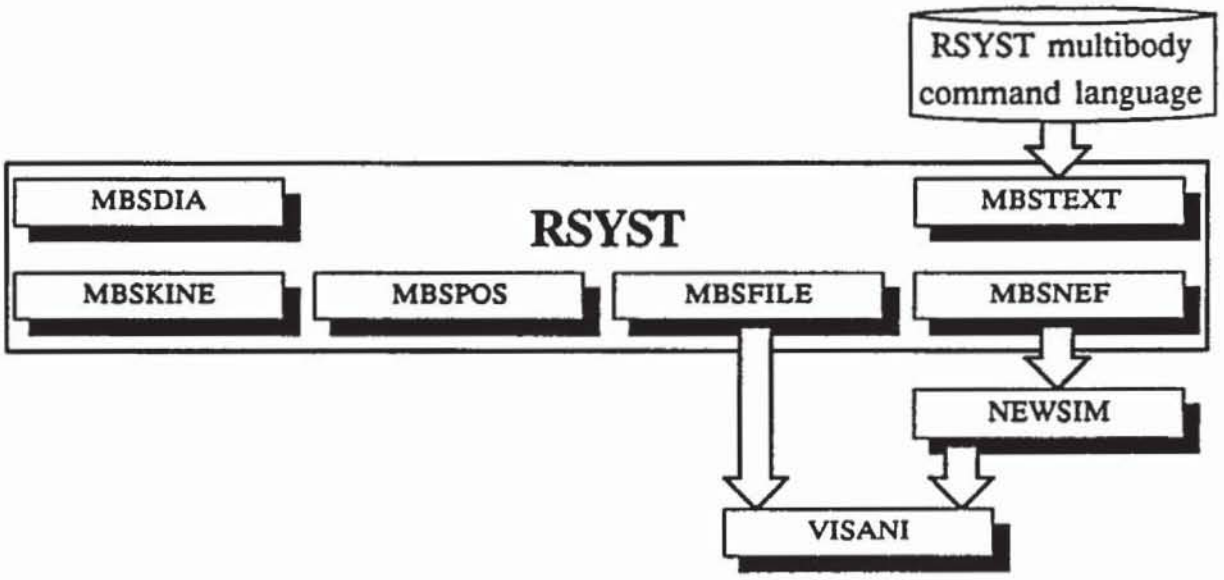

Figure 8: CAD-3D modeling RSYST interfaces to dynamics modules NEWEUL, NEWSIM and visualization modul VISANI

4 Generation of equations of motion starting from the database

The generation of equations of motion and the embedding of these equations to simulation software is especially in case of large multibody models very time consuming and prone of errors. Starting from the description of the multibody system stored on the database, the modul NEWEUL [9] generates symbolic equations of motions and

all information necessary for the automatic simulation. The modul NEWSIM [10] uses in the next step the compiled symbolical equations of motion for the simulation. Using the object-oriented datamodel the modules NEWEUL and NEWSIM are tools of a modular software package of the multibody system approach, see Figure 9.

In a first step the information stored in the database has to be extracted. In a modular concept the generation of equations of motion and the simulation have to be separated. The datamodel includes all the information neccessary for the generation of the equations of motion and, an adapted version of NEWEUL can be used as module in the database concept. Based upon a Newton-Euler formalism the symbolical equations of motion are generated using d'Alembert's or Jourdain's principle to eliminate the reactions forces and torques, see Schiehlen [24]. By means of a special, for the multibody system approach developed formulamanipulator, it is possible to generate the equations of motion with minimal costs of computation time, see Kreuzer [8]. The symbolical equations of motion can be used on the one hand in the simulation environment NEWSIM and on the other hand in any general purpose simulation environment, e.g. ACSL [1] or DSSIM [18].

At first, from the objects *interact* and *joint* the topology of the multibody system is computed. Additionally from the object *joint* the generalized coordinates are determined. The kinematical description of multibody systems is done by the definition of frames relatively to any arbitrary frame. These frames define rigid bodies, joints, auxiliary frames, and reference frames, too. Additionally the mass-geometric properties and the applied forces and moments are neccessary. These data can be found in the objects *interact* and *force*, see Figure 9.

The modul NEWSIM serves for the numerical simulation of the generated symbolic equations of motion. It is easy to study the influence of parameters or to optimize the dynamical behaviour with respect to some specified criteria. NEWSIM has the possibility to treat additional differential or differential-algebraic equations. For integration in the time-domain different integration schemes are e.g. Runge-Kutta methods, Adams-methods, BDF-methods. For multibody systems including closed loops a modified Adams-Bashforth-Moulton method is implemented, see Leister [11]. All neccesary routines for the automatic simulation software are generated by NEWEUL, Figure 10. After the compilation and binding step the problem-specific programm takes all parameters and options from the datafile. This program reads all options, initial conditions, fixed system parameters like masses, moments of inertia, geometric data, stiffness constants, and further data from the input file and solves the equations of motion of the problem.

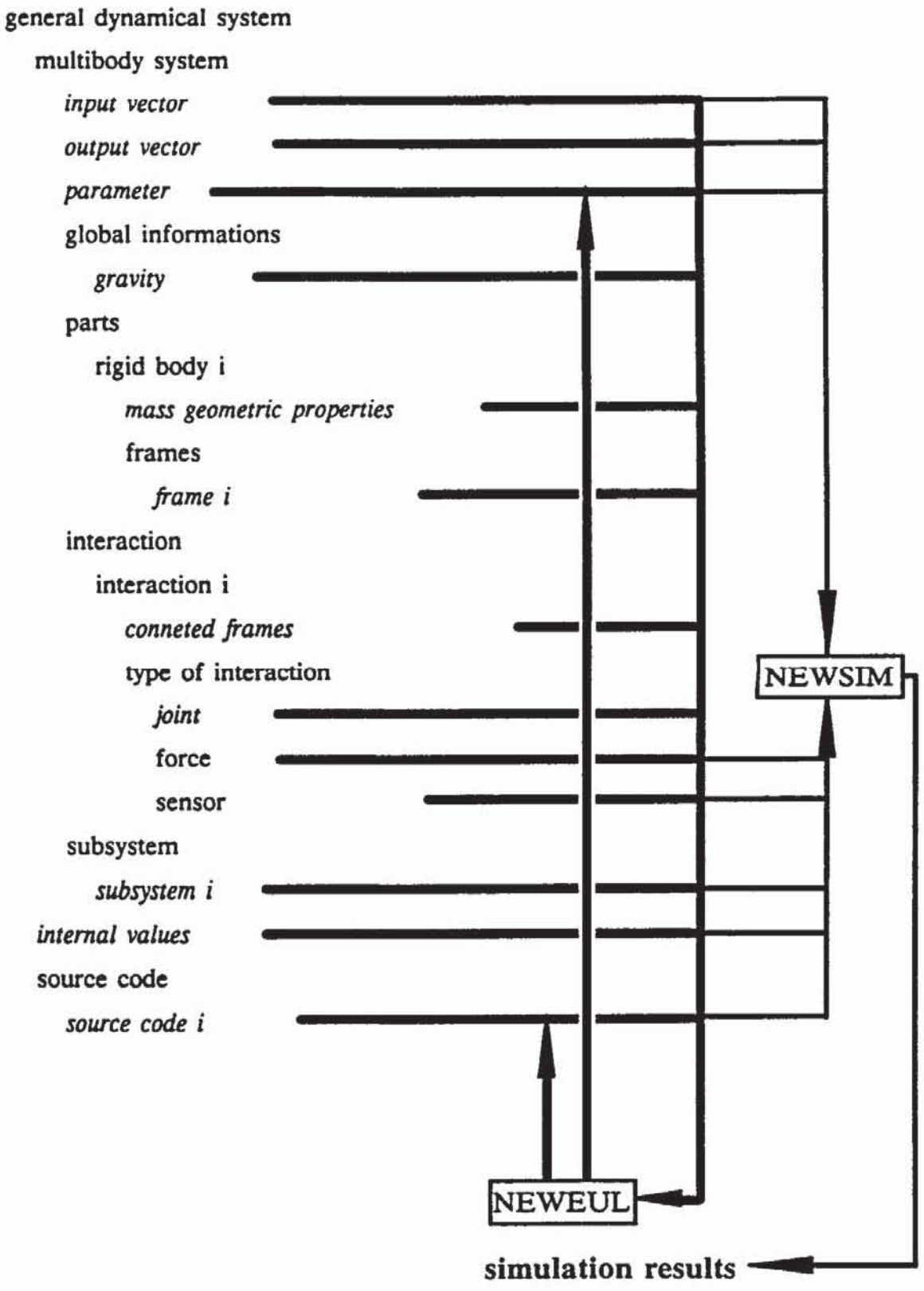

Figure 9: Dataflow of the datamodel

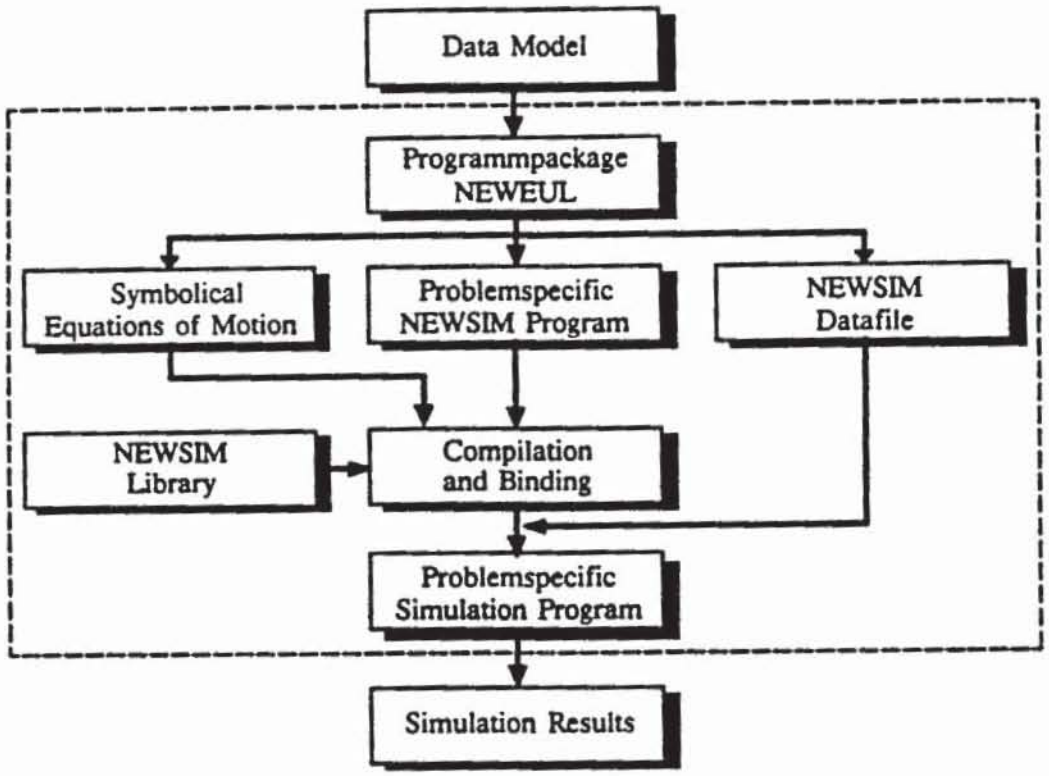

Figure 10: Simulation of the dynamic behaviour with NEWEUL and NEWSIM

5 Formalism for multibody systems including loops

Modeling dynamical systems by the method of multibody systems results in either ordinary differential equations (ODEs) using minimal coordiantes or coupled differential and algebraic equations using cartesian and redundant coordinates (DAEs). Often ODEs are integrated numerically by explicit multistep integration algorithms whereas DAEs have to be integrated by implicit or halfimplicit methods. Numerical experiments have shown, Leister [11], that the integration algorithms for ODEs seems to be more efficient than algorithms for DAEs. Thus, it is advantageous to describe multibody systems by a minimal number of pure differential equations, the so-called state space form.

For multibody systems with closed loops the use of minimal coordinates is not always convenient. Closed loops can be cut up for describing the system kinematically, see Figure 11. Supposed that the open-loop system has n degrees of freedom, one has to choose a set of n generalized coordinates $\boldsymbol{x} \in I\!R^n$ resulting in ordinary differential equations of motion for the partially unconstrained system. The closed-loop system will then have $f = n - m$ degrees of freedom. The dimension of the equations of

motion can be further reduced to the number of degrees of freedom of the closed-loop system. Such a reduction to the state space form can be achieved by several methods.

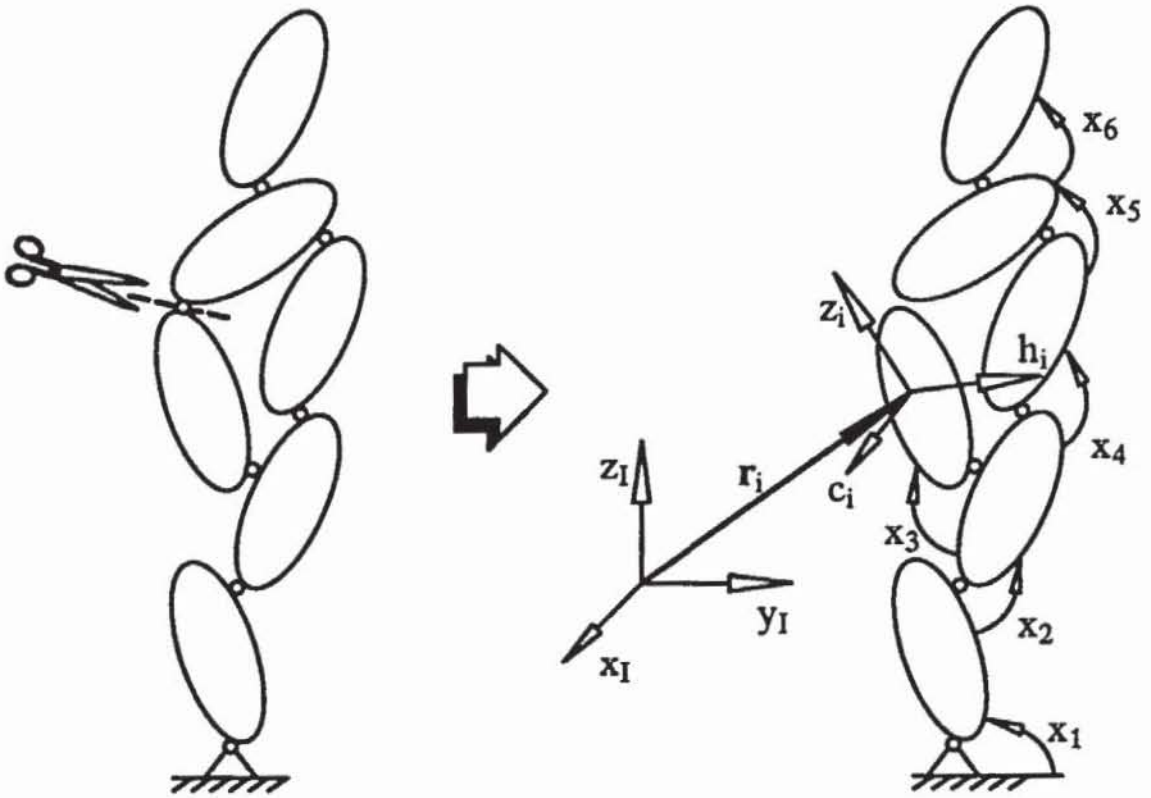

Figure 11: Cutting up closed-loop systems

The coordinate partitioning method, e.g. Wehage and Haug [30], locally uses f of the generalized coordinates as independent and the remaining coordinates as dependent coordinates. Then, the equations of motion and the constraints can be decomposed, the Lagrange multipliers can be eliminated from the equations, and one ends up with pure differential equations. This procedure has to be carried out numerically at every time step and, therefore, is very time consuming.

The same procedure can also be performed symbolically, e.g. [8]. The user has to make an a priori choice of independent coordinates, the dependent variables are declared as auxiliary variables. For example the formalism NEWEUL [9] is able to generate direcely symbolical equations of motion in state space form. In general, the choice of independent variables is sometimes not valid for the whole time domain of interest. Thus, the simulation code has to switch between several different forms of equations of motion for avoiding singularities. It is obvious that it is difficult to automate such a procedure for general dynamic systems.

The independent coordinates need not be part of the generalized coordinates $\boldsymbol{x}$ of the open loop system can also be chosen more general as a linear combination of the generalized coordinates to make the problem well conditioned, e.g. [7], [13]. But, these methods are even more time consuming than the already mentioned coordinate partitioning method.

Here the advantages of these methods are combined: the efficiency of symbolical equations of motion in state space form, the generality of the numerical algorithms which can choose new independent variables at every time step, and the good condition of a free choice of independent coordinates.

In a first step the loops of the multibody system have to be cut up, see Figure 11. The resulting tree can be described by generalized coordinates $\boldsymbol{x} \in \mathbb{R}^n$. Additionally, a set of coordinates $\boldsymbol{y} \in \mathbb{R}^f$ the number which is minimal is introduced, which need not to be specified in this early stage of modeling. These minimal number of coordinates are by definition independent and describe fully the kinematics of the closed-loop system, whereas the generalized coordinates $\boldsymbol{x}$ depend on $\boldsymbol{y}$ and time, i.e. $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{y}, t)$. For those parts of the closed-loop system which have tree structure, coordinates x_i and y_j may be identical, generalized coordinates associated with independent closed loops will depend only on the minimal number of coordinates which describe the corresponding loop. For example, in Figure 11 one has $n = 6$ and $f = 4$ with the coordinates $x_1 = y_1$, $x_2 = y_2$, $x_3 = x_3(y_3)$, $x_4 = x_4(y_3)$, $x_5 = x_5(y_3)$, $x_6 = y_4$. In the next step, the location of the center of gravity and the orientation of each body i has to be described by the position vector $\boldsymbol{r}_i$ and the rotation tensor $\boldsymbol{S}_i$,

$$\begin{aligned} \boldsymbol{r}_i &= \boldsymbol{r}_i(\boldsymbol{y}, \boldsymbol{x}(\boldsymbol{y}, t), t), \\ \boldsymbol{S}_i &= \boldsymbol{S}_i(\boldsymbol{y}, \boldsymbol{x}(\boldsymbol{y}, t), t), i = 1(1)p. \end{aligned} \tag{3}$$

where p is the number of bodies. Although an explicit dependence of $\boldsymbol{r}_i$ and $\boldsymbol{S}_i$ on $\boldsymbol{y}$ would not be necessary it was included for increasing the efficiency of the proposed algorithm by taking identities like $x_i = y_j$ into consideration. Differentiation with respect to time yields velocity $\boldsymbol{v}_i$ and angular velocity of $\boldsymbol{\omega}_i$ each body:

$$\begin{aligned} \boldsymbol{v}_i &= \frac{d\boldsymbol{r}_i}{dt} \\ &= \left[\frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{y}} + \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}}\right]\dot{\boldsymbol{y}} + \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial t} + \frac{\partial \boldsymbol{r}_i}{\partial t} \\ &=: \boldsymbol{J}_{Ti}(\boldsymbol{y}, \boldsymbol{x}, \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}}, t)\dot{\boldsymbol{y}} + \bar{\boldsymbol{v}}_i(\boldsymbol{y}, \boldsymbol{x}, \frac{\partial \boldsymbol{x}}{\partial t}, t), \\ \boldsymbol{\omega}_i &= \boldsymbol{J}_{Ri}(\boldsymbol{y}, \boldsymbol{x}, \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}}, t)\dot{\boldsymbol{y}} + \bar{\boldsymbol{\omega}}_i(\boldsymbol{y}, \boldsymbol{x}, \frac{\partial \boldsymbol{x}}{\partial t}, t) \end{aligned} \tag{4}$$

where $\boldsymbol{J}_{Ti}$ and $\boldsymbol{J}_{Ri}$ are the Jacobians for the translational and rotational motions. By a second differentiation the accelerations are obtained. With d'Alembert's principle

by premultiplication with the transposed Jacobian, we obtain the ODE-form of the equations of motion:

$$M(\boldsymbol{y},\boldsymbol{x},\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}},t)\,\ddot{\boldsymbol{y}} \;+\; \boldsymbol{k}(\boldsymbol{y},\dot{\boldsymbol{y}},\boldsymbol{x},\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}},\frac{\partial \boldsymbol{x}}{\partial t},\frac{d}{dt}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}},\frac{d}{dt}\frac{\partial \boldsymbol{x}}{\partial t},t) = \boldsymbol{q}(\boldsymbol{y},\dot{\boldsymbol{y}},\boldsymbol{x},\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}},\frac{\partial \boldsymbol{x}}{\partial t},t) \tag{5}$$

where $\boldsymbol{M}$ is the matrix of inertia, $\boldsymbol{k}$ is the vector of centrifugal and Coriolis forces, and $\boldsymbol{q}$ is the vector of applied forces.

Obviously, the accelerations $\ddot{\boldsymbol{y}}$ cannot be computed from equation (5) alone if the state $\boldsymbol{y}$, $\dot{\boldsymbol{y}}$ is given, only. One needs more equations specifying $\boldsymbol{x}$ and its derivatives. For specifying $\boldsymbol{x}$ the constraints have to be accomplished by (n-m) further equations $\boldsymbol{\Psi} = \boldsymbol{0}$, i.e.

$$\boldsymbol{g}(\boldsymbol{y},\boldsymbol{x},t) := \begin{bmatrix} \boldsymbol{c}(\boldsymbol{x},t) \\ \boldsymbol{\Psi}(\boldsymbol{y},\boldsymbol{x}) \end{bmatrix} = \boldsymbol{0}, \tag{6}$$

such that the Jacobian $\boldsymbol{G} := \partial \boldsymbol{g}/\partial \boldsymbol{x}$ is nonsingular. Then the solution $\boldsymbol{x}$ of (6) will locally be unique. Although equations (6) are used for determining the generalized coordinates $\boldsymbol{x}$, a special choice of functions $\boldsymbol{\Psi}$ has to be regarded as a choice of the minimal number of coordinates $\boldsymbol{y}$. This becomes more clear from the fact that the coordinates $\boldsymbol{x}$ already have physical meaning from describing the kinematics of the open-loop system, whereas $\boldsymbol{y}$ has never been specified explicitly. Subsequently a special choice for $\boldsymbol{\Psi}$ will be made.

Since equation (6) holds for all $\boldsymbol{y}$, differentiation with respect to $\boldsymbol{y}$ and with respect to time t yields relations for the computation of $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}}, \frac{\partial \boldsymbol{x}}{\partial t}, \frac{d}{dt}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}}, \frac{d}{dt}\frac{\partial \boldsymbol{x}}{\partial t}$, see [12]. In principle, the equations of motion are now solvable. The coordinates $\boldsymbol{x}$ and their derivatives may be regarded as substitution variables which can be calculated for a given state $\boldsymbol{y}$, $\dot{\boldsymbol{y}}$. Then, the ordinary differential equations (5) can be integrated numerically by any general purpose integration algorithm. The solution of equations (6) can be found iteratively by the Newton-Raphson method.

Several systems of algebraic equations with the coefficient matrix

$$\boldsymbol{G} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}} = \begin{bmatrix} \boldsymbol{C} \\ \\ \dfrac{\partial \boldsymbol{\Psi}}{\partial \boldsymbol{x}} \end{bmatrix} \tag{7}$$

have to be solved. For avoiding singularity of $\boldsymbol{G}$ the rows of $\partial \boldsymbol{\Psi}/\partial \boldsymbol{x}$ have to be linearly independent of the rows of $\boldsymbol{C}$. But starting from the Jacobian $\partial \boldsymbol{\Psi}/\partial \boldsymbol{x}$ the functions $\boldsymbol{\Psi}$ would have to be found by integration which is not possible in

general. Therefore, the minimal number of coordinates is simply chosen as a linear combination of $\boldsymbol{x}$:

$$\boldsymbol{\Psi} := \boldsymbol{K}\boldsymbol{x} - \boldsymbol{y} \;=\; \boldsymbol{0} \tag{8}$$

where $\boldsymbol{K}$ is a constant $f \times n$ coefficient matrix.

The best choice of the coordinates $\boldsymbol{y}$ is to make the row space of $\boldsymbol{K}$ orthogonal to the row space of $\boldsymbol{C}$ for the actual time and state. This can be achieved for instance by a singular value decomposition of $\boldsymbol{C}$. The condition of $\boldsymbol{G}$ can further be improved by normalizing the rows of $\boldsymbol{C}$ and $\boldsymbol{K}$.

The coordinates $\boldsymbol{y}$ have been chosen as a linear combination of the generalized coordinates with a constant coefficient matrix $\boldsymbol{K}$. Although this may have been a good choice at the initial time point, the Jacobian (7) can become singular as simulation goes on. Since the constraint Jacobian $\boldsymbol{C}$ depends on state and time, some of its rows may become linear dependent on the rows of $\boldsymbol{K}$.

It is important to have a criterion monitoring such singularities. The norm of the Jacobian $\partial\boldsymbol{x}/\partial\boldsymbol{y}$ is well suited for this purpose, see [11]. For changing the coordinates $\boldsymbol{y}$ a new matrix $\boldsymbol{K}$ has to be chosen. It is best to use as $\boldsymbol{K}$ an orthogonal basis of the nullspace of $\boldsymbol{C}$ at the actual position. This will keep the rows of $\boldsymbol{K}$ independent of the rows of $\boldsymbol{C}$ for a large range of simulation time. After a change of minimal coordinates a multistep integration algorithm cannot continue because the information belonging to the past time points is not consistent with the new choice. A restart of the integration procedure would cause a loss of accuracy and efficiency due to reduced time step size and order. Thus, it is useful to transform the internally stored information on the polynomials which are used for prediction of the state at the next time step. Since the Adams-Moulton formulas are linear, the same transformation rules apply to the polynomial coefficients of $\boldsymbol{x}$ which are stored for integrating numerically.

6 Visualization of simulation results

A convenient verification a dynamic visualization of a multibody system simulation is obtained by a 3D computer graphics animation. Animation methods differ according to the geometry model, rendering algorithms and possible user interaction. The most sophisticated animation method is achieved by rendering algorithms like raytracing and radiosity. These rendering techniques result in realistic images, but suffer from time-consuming computations. During image display, no interactive modification of the view projection is possible. A raytraced image of the crank-slider mechanism is shown in Figure 12.

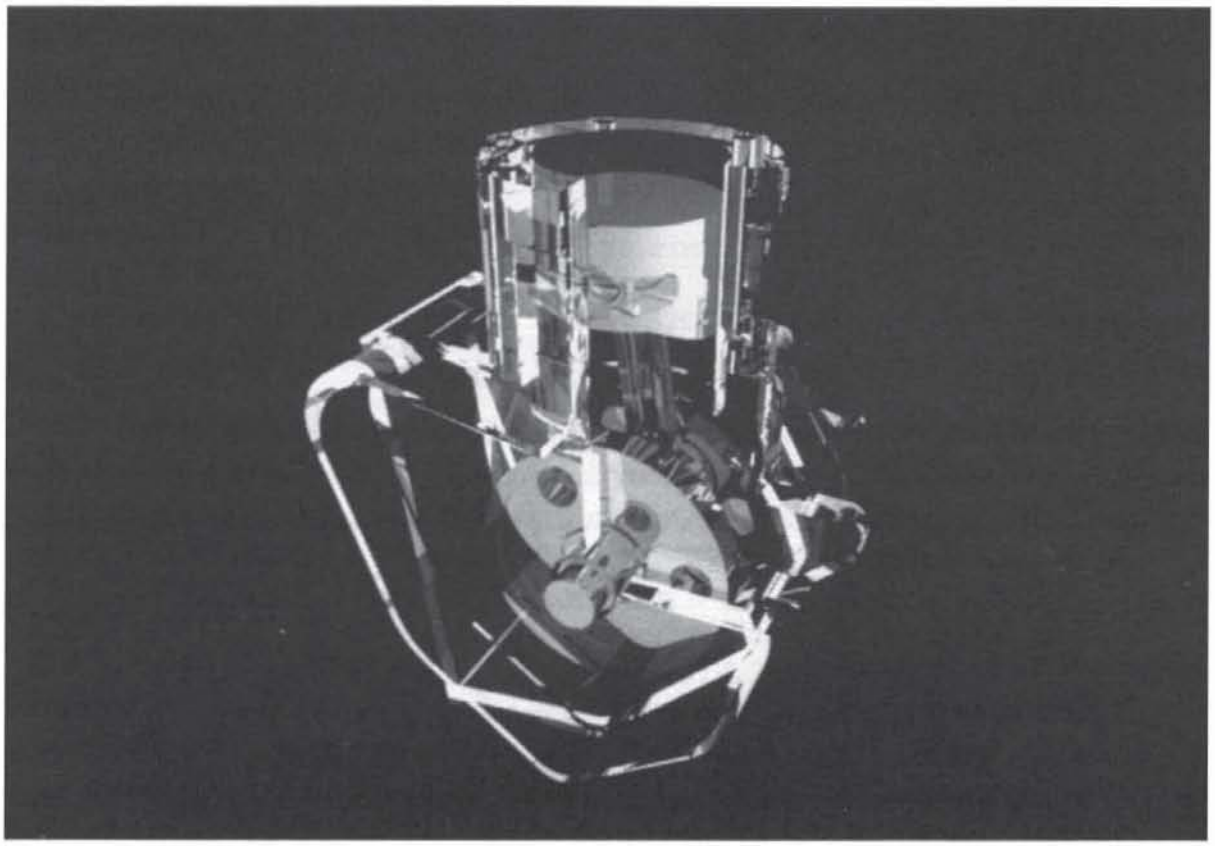

Figure 12: Raytracing of crank slider mechanism

Most CAD-3D-systems offer modules for the generation of images with hidden line and hidden surfaces removal and shaded surfaces. Often, the solid model and rendering algorithms yield sophisticated 2D drawings for documentation purposes, but allow a dynamic visualization only in a wireframe mode.

Consequently the unified approach to display a broad variety of simulation result for different initial conditions, visualization systems and applications is based on the planar face model, see Figure 3. The visualization module VISANI for the interactive, high speed animation of arbitrary multibody systems is described by Daberkow [3]. As a result of the simulation, a time plot of the crankshaft bearing force of the mechanism under an applied piston gas force and an animated sequence is shown in Figure 13.

7 Conclusion

In this paper an integrated CAD-3D modeling, simulation and visualization of multibody system dynamics is introduced. A unified general data model including the graphic description is presented. To support the preceding CAD-3D-modeling stage, a unified spatial graphic representation for multibody elements is designed. Object-

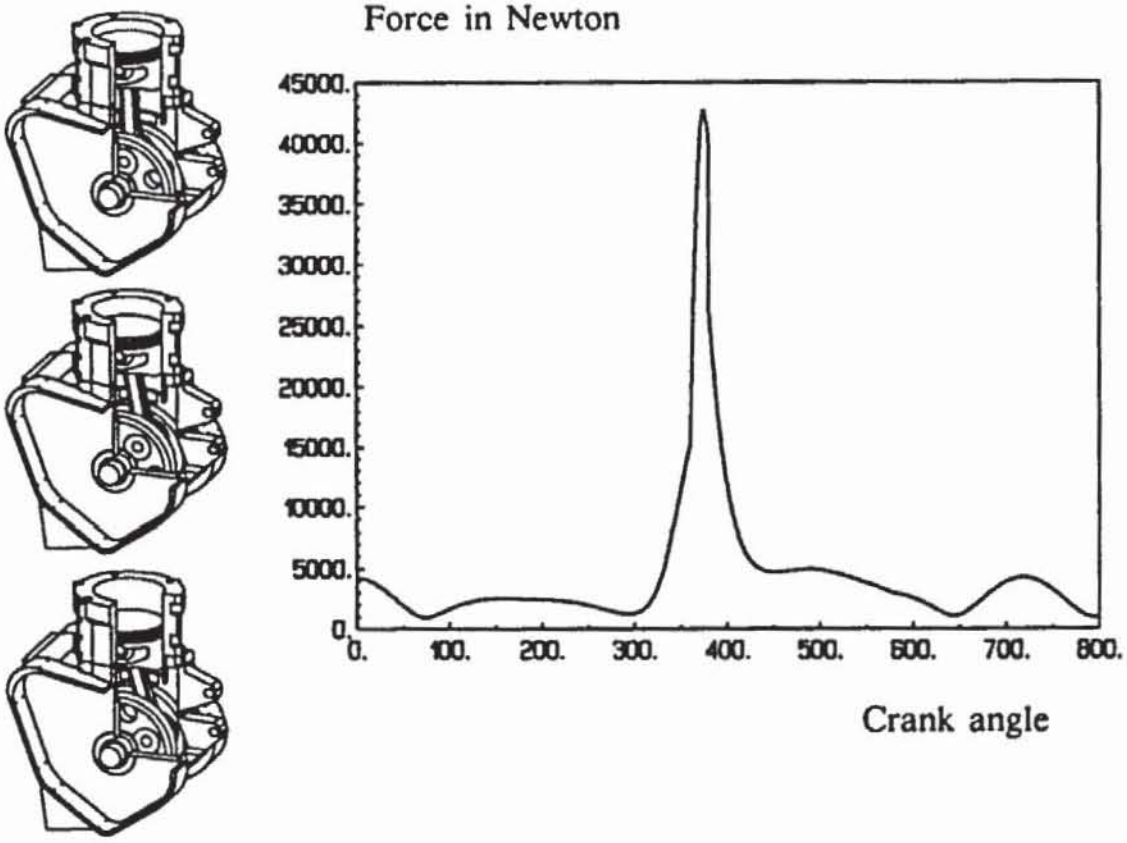

Figure 13: Time plot and animated sequences of the crank slider mechnism

oriented classes and operations are then implemented in a system independent multibody modeling kernel library and integrated in a commercial CAD-3D system. A crank slider mechanism serves as an example to demonstrate the interface from modeling to a data base system. From the multibody model data base, an integrated Newton-Euler formalism generates a set of symbolical ordinary differential equations, which are solved by explicit multistep integration algorithms. Thereby, a minimal set of generalized coordinates is specified during numerical integration without restart of the integration algorithm. The final visualization of the crank slider mechanism demonstrates that this integrated approach fits the criteria of a modular, automated design and simulation environment.

References

[1] ACSL-Advanced Continuous Simulation Language Reference Manual. Inc. Concord/Mass.: Mitchell u. Gauthier Assoc., 1987.

[2] ARIES Conceptstation Software Simulation Mechanism Reference. Aries Technology Inc., Lowell, MA, 1990.

[3] Daberkow, A.: Zur CAD-gestützten Modellierung von Mehrkörpersystemen. Ph.D. Dissertation, Stuttgart, 1992.

[4] Haug, E.J.: Computer Aided Kinematics and Dynamics of Mechanical Systems. Allyn and Bacon, Boston, MA, 1989.

[5] Hocke, M.; Rühle, R.; Otter, M.: An Open Software Environment for the Analysis and Design of Multibody Systems. Appears in this volume.

[6] Hollar, M.G.; Rosenthal, D.E.: Concurrent Design and Analysis of Mechanisms. Rasna Corporation, San Jose, CA, 1991.

[7] Kim, S.S.; Vanderploeg, M.J.: QR Decomposition for State Space Representation of Constrained Mechanical Dynamic Systems, ASME Journal of Mechanisms, Transmissions, and Automation in Design, Vol. 108, 1986, pp. 183 - 188.

[8] Kreuzer, E.: Symbolische Berechnung der Bewegungsgleichungen von Mehrkörpersystemen, Fortschr.-Ber. der VDI-Zeitschriften, Reihe. 11, Nr. 32, Düsseldorf: VDI-Verlag, 1979.

[9] Kreuzer, E.; Leister, G.: Programmsystem NEWEUL'90, Anleitung, Stuttgart: Universität, Institut B für Mechanik, AN-24, 1991.

[10] Leister, G.: Programmpaket NEWSIM. Stuttgart: Universität, Institut B für Mechanik, AN-25, 1991.

[11] Leister, G.: Beschreibung und Simulation von Mehrkörpersystemen mit geschlossenen kinematischen Schleifen. Fortschr.-Ber. der VDI-Zeitschriften, Reihe 11, Nr. 167. Düsseldorf: VDI-Verlag, 1992.

[12] Leister, G.; Bestle, D.: Symbolic-numerical Solution of Multibody Systems with Closed Loops. Vehicle System Dynamics, Vol. 21, pp. 129-142, 1992.

[13] Mani, N.K.; Haug, E.J.; Atkinson, K.E.: Application of Singular Value Decomposition for Analysis of Mechanical System Dynamics, ASME Journal of Mechanisms, Transmissions, and Automation in Design, Vol. 107, 1985, pp. 82 - 87.

[14] Meyer, B.: Object-oriented Software Construction. Prentice Hall, New York, 1988.

[15] Mortenson, M.E.: Geometric Modeling. John Wiley, New York, 1985.

[16] Orleanda, N.: Node-Analogous Sparsity-Oriented Methods for Simulation of Mechanical Systems. Ph.D. dissertation, University of Michigan, 1973.

[17] Otter, M.; Hocke, M.; Daberkow, A.; Leister, G.: Ein objektorientiertes Datenmodell zur Beschreibung von Mehrkörpersystemen unter Verwendung von RSYST. Stuttgart: Universität, Institut B für Mechanik, IB-16, 1990.

[18] Otter, M.; Gaus, N.: ANDECS-DSSIM: Modular Dynamic Simulation With Database Integration. User's Guide, Version 2.1. Oberpfaffenhofen: DLR, TR R50-91, 1991.

[19] Otter, M.; Hocke, M.; Daberkow, A.; Leister, G.: An Object Oriented Datamodel for Multibody Systems. Appears in this volume.

[20] Pahl, G.: Konstruieren mit 3D-CAD Systemen: Grundlagen, Arbeitstechnik, Anwendungen. Springer, Berlin, 1990.

[21] PARASOLID Solid Modeling System Kernel Interface Reference Manual. Shape Data Ltd., Cambridge, England, 1990.

[22] Roberson, R.E., Schwertassek, R.: Dynamics of Multibody Systems. Springer, Berlin, 1988.

[23] Rühle, R.: RSYST, ein integriertes Modulsystem mit Datenbasis zur automatischen Berechnung von Kernreaktoren. Stuttgart: Universität, IKE 4-12 1973.

[24] Schiehlen, W.: Technische Dynamik. Stuttgart: Teubner Verlag, 1986.

[25] Schiehlen, W. (ed): Multibody Systems Handbook. Berlin/...: Springer–Verlag, 1990.

[26] Schiehlen, W.O., Daberkow, A.: Modeling, Simulation and Animation of Nonlinear Multibody Systems. Proc. of the 3. Conference of Theoretical and Applied Mechanics, Academy of Scientific Research and Technology, Cairo, pp. 27-48, 1988.

[27] Shampine, L.F.; Gordon, M.K.: The Computer Solution of Ordinary Differential Equations; the Initial Problem, Freeman, San Francisco, 1975.

[28] SIGRAPH-CAD-3D. SIEMENS NIXDORF AG, München, 1992.

[29] Rosenthal, D.E., Sherman, M.A.: High performance multibody simulation via symbolic equation manipulation and Kane's method. Journal of Astronautical Sciences, 34, pp. 223-239, 1986.

[30] Wehage, R.A.; Haug, E.J.: Generalized Coordinate Partitioning for Dimension Reduction in Analysis of Constrained Dynamic Systems, ASME Journal of Mech. Design, Vol. 104, 1982, pp. 247 - 255.