A FLEXIBLE ELECTRONIC CONTROLLER FOR A MANIPULATOR-TYPE ROBOT

Patrick J. Gibbs, F. Jones and J.V. Ringwood

School of Electronic Engineering,
Dublin City University,
Glasnevin, Dublin 9.

ABSTRACT

Manipulator arm construction has changed little over the decades and is unlikely to change radically in the near future. The mechanical design necessary to achieve dexterity results in a system with complex dynamic properties. However, many manipulator manufacturers choose to ignore this complexity, concentrating on the mechanical design aspects rather than the design of the dynamic controller. In most cases, simple fixed-parameter single-loop PID compensators are utilised. In spite of the fact that the compensators are implemented on programmable devices, there is simply not enough processing power available to implement an improved dynamic control strategy.

A multiprocessor controller has been developed which allows all the hierarchical levels of a manipulator controller to be implemented. The major advantage of the new controller is its ability to handle complex and time consuming dynamic algorithms for positioning of the robot end effector. This has been accomplished by adopting a master/slave multiprocessor configuration comprising a 20 MHz IBM PC/AT (80386) with a number of DSP cards based around the NEC 77230 floating-point DSP chip. Analog and digital input/output interfaces are provided for reading position signals and providing command signals.

The motivation for the provision of such a controller was the desire to implement linear and nonlinear self-tuning control strategies. Both centralised (multivariable) and decentralized (single-loop) control strategies are considered and the new controller caters for both schemes by virtue of (a) the master/slave configuration with individual DSP boards for each joint, and (b) inter-board communications, allowing joint interactions to be catered for.

In the paper, some of the identification algorithms required to support the nonlinear self-tuning strategies are described and real-time results presented. These results demonstrate the operation of the new controller and indicate some of its capabilities.

ROBOT CONTROL SYSTEMS

Commercial robot systems are generally restricted in terms of modifications to hardware and software for real time control. This may be acceptable in workspaces where the repetition of a limited sequence of motions is all that is required. In both flexible manufacturing and in robotic research environments, however, the primary considerations are ease of modification, adaptability and programmability. These three characteristics are essential in order to manufacture a new product for the evaluation of a new sensor system or robot control algorithm.

NEW ROBOT CONTROL SYSTEM

The PUMA 560, because of its two distinct hardware levels, offers what is known as decentralized control structure. Such structures have been widely accepted [3] by the robotics industry due to ease of implementation and tolerence of failure. The main advantage, however, of such a structure is that it allows for easier implementation of the control layers discussed. For this reason, it was decided that the new hardware structure should be a predominantly decentralized one. Also the new control structure should offer the following:

1) floating point processors to perform mathematical calculations with high precision and at high enough speed for real-time control,
2) interfacing hardware which is compatible with the existing unimation hardware,
3) software that can be written in a single high-level language,
4) a memory capacity suitable for large program storage,
5) an ability to implement multivariable control via interprocessor communication.
6) the ability to provide real-time path planning.
7) the ability to connect sensory devices through serial, parallel or bus interfaces.

Finally, on top of all these requirements the new control structure must be economically viable. Otherwise it is not a realistic alternative to the existing control structure as far as the robot manufacturer is concerned.

Hardware Options

Numerous implementations of the control structure's upper level have replaced the existing upper level computer with a more powerful central computer. In one example the LSI-11/02 was replaced by the more powerful LSI-11/23 in conjunction with a Microvax. This combination provides the user with full floating-point capabilities, high-level language capabilities and an abundance of memory space. The implementation of such a system effectively doubles the cost of the original system [4], making it economically impractical. More recent implementation such as the TUNIS and SIERA have replaced the existing upper level with powerful personal computers (PCs). Both of these systems are capable of offering the capabilities just mentioned above but at a fraction of the previous cost. For this reason it was decided to use a personal computer to implement the new upper level.

System Description

The personal computer chosen was an Intel-based 80386 IBM compatible personal computer. The features which governed the choice of this personal computer included the presence of :

1) a 32-bit architecture (data and addressing),
2) a clock speed of 20 MHZ,
3) the ability to add a floating-point coprocessor (80387),
4) 1 megabyte of RAM,
5) an 80 megabyte harddisk and
6) seven parallel expansion slots.

Most commercial robots, like the PUMA 560, are sold with a dedicated programming language which run on a dedicated hardware configuration. As a result, the characteristics mentioned above are not present in the PUMA 560. This necessitates the design of a new more flexible controller for this robot. Shortcomings can occur in three main areas: the controller software; the controller hardware and in the control algorithm used to control the robot.

The Unimation Control System

In the case of the PUMA 560 industrial robot, a limited form of task-space control is provided by VAL2 (Victor's Assembly Language). As an operating system, VAL provides the necessary input/output to control the robot, retrieve data from the floppy disk and to interact with the user via the terminal or a teaching pendant. Despite the relative ease of use and its capabilities, the VAL based system is seriously lacking in terms of flexability, expandability and is devoid of the ability to implement powerful real-time task space control.

The Unimation control hardware [2] consists of an LSI-11/02, and six Rockwell 6503 microprocessors each with a digital–to–analog converter (DAC), a current amplifier and joint position feedback sensors. The hardware is hierarchically arranged. The upper level of the system hierarchy consists of the LSI-11/02 microcomputer which serves as a supervisory computer, while the lower level of the hierarchy consists of the 6503 Rockwell µPs and the remaining hardware just mentioned.

This PUMA 560 hardware suffers from some limitations. These have been described by Goldenberg [1] :

1) both levels of the controller hierarchy contain only fixed point processors,
2) the existing memory in both levels is inadequate to support large programs,
3) the instruction speed of the Rockwell 6503 µPs and the LSI-11/02 are inadequate to implement computationally complex control algorithms and finally,
4) it is impossible to add additional sensors to the robot, such as vision and tactile sensors, without a complete redesign of the lower level.

From this list of limitations it can be seen that if a more flexible hardware control structure, capable of implementing complex real time control, is required, then the existing Unimation controller hardware must be replaced with a more flexible alternative.

The PUMA 560 controller is basically a position plus derivative control method. One of the main disadvantages of such a method is that the feedback gains are constant and prespecified. It does not have the capacity to update the feedback gains under varying payloads. An industrial robot such as the PUMA 560 is a highly non-linear system. One can see that these nonlinearities are due to inertial, gravity and other coupling effects. As a result the positions , velocities and accelerations of the PUMA's joints are dependent on the magnitude and variations in these effects. This control algorithm, with it's fixed feedback gains, fails to take these variations into account. In fact, the PUMA moves with noticable vibrations at reduced speed [1] because of the controller gains being too high. This makes the robot suitable only for performing simple pick and place tasks that do not have a great deal of precision. To improve the performance of the robot it is, therefore, necessary to replace this control algorithm with one that is capable of tracking some or all of nonlinearities present. This algorithm should also be implemented using floating-point arithmethic to achieve higher precision.

From this list of features it can seen that the new upper level offers a development and storage environment suitable for large program generation. It also offers a fast execution speed for such programs, even if they contain floating-point calculations. The expansion slots offer the ability to add extra memory and the ability to interface the new lower level.

To replace the lower level of the controller architecture it was necessary again to choose a processor with high speed floating-point capabilities. One option considered was the option chosen by Goldenberg [1]. This implementation uses a PC to implement both levels. This means that the tasks of the upper and lower levels have to be executed serially and not in parallel like the existing control structure. Considering, the high speed sampling required for robot control, this serial execution of tasks limits the complexity of both the upper and lower level tasks. One solution which has become more popular in recent years is to use advanced signal processors (ASPs) to implement this level. The reasons for their rise in popularity include [5] the reduction in operation and development time which they offer. As well as this, recent advances in VSLI technologies have meant cheaper ASP chips.

For these reasons, it was decided to use an ASP configuration to implement the lower level of the controller. The ASP chosen for this level was the NEC µPD77230 [6]. The µPD77230 is capable of processing digital signals at high speeds and with good accuracy. It can execute arithmethic operations with 32-bit, floating point data (8 bits for exponent and 24 bits for mantissa) or 24-bit, fixed-point data at 150 ns/instruction. Its internal circuitry comprises of a multiplier (32 x 32 bits), an ALU (55 bits), an instruction ROM (1K by 32 bits) and one pair of data RAM pointers (512 words by 32 bits each). The µPD77230 can operate in two modes : master or slave. By operating in master mode the processor's instruction area occupies 8K words by 32 bits of memory. In addition, it allows for three stage pipelining and provides a dedicated data bus for internal RAM, a multiplier and an ALU. Such an arrangement makes the processor suitable to process algorithms in which a few operations (such as addition of terms) occur repeatedly. These are the type of operations that occur in the more complex control algorithms such as the computed torque method [7]. In [6] it was found that a single µPD77230 was capable of achieving throughput rates of 1,350 setpoints per second and by utilizing the pipelining nature fully it was found that this algorithm could achieve a throughput of 2,220 setpoints per second. These figures produce controller sampling rates of 0.740 mS and 0.450 mS respectively. These sampling rates are much faster than the existing controller which implements a much simpler PD control algorithm. These timing statistics, coupled with the fact that the computed torque method is one of the most computationally complex robot control algorithms means that a µPD77230-based lower level is well capable of implementing real-time control algorithms for robotic control.A block diagram of the system is shown in Fig.1.

Since the new hardware configuration is a hierarchical, multi-processor system, and as a result it requires a considerable amount of inter-processor communication to perform its robot control function. Fortunately, since the two levels in the new PUMA 560 controller are "off-the-shelf" items, use can be made of existing software tools to achieve the inter-processor communication desired.

This type of robot control hardware, with a personal computer as the upper hardware level, allows for easier implementation in both industrial and educational environments. This is due to the general familarity with the personal computer operating system and hardware. By using a commercially available operating system (MSDOS) with the robot control hardware, one can speed up the development process and the learning curve of potential users, since features such as file management, batch file generation and on line debugging tools are available.

Performance

The calculation functionality of the new hardware can be defined in terms of the speed at which the basic operations such as add, subtract, divide and multiply can be performed on fixed and floating-point data. For the personal computer the fixed point operations were found to take three clock cycles to execute (i.e., 150 ns). Double precision floating point additions were found to take 10μs and multiplications approximately 32 μs each.

In the lower level computational functionality involves μPD77230 board's ability to perform floating and fixed point addition, subtraction, division and multiplication. For fixed point data these calculations were found to take one instruction cycle or 150ns. In the floating-point case, addition and subtraction take five instruction cycles, with multiplication taking six instruction cycles. This means that the lower level is capable of performing thousands of additions and multiplications in one millisecond. The advantage can be seen more clearly if one examines the algorithms developed in [8],[9] and [10]. These algorithms are among some of the most computationally complex available, yet prelimenary calculation suggest that these algorithms can be implemented in real-time using the μPD77230 boards. In the case of [8] and [9] these calculations show that both could be implemented in times less than 0.5 ms, while [10] could be implemented in a time which is less than 0.8 ms. The same algorithms, if implemented on the existing Rockwell 6503 μPs would require that the sampling interval be increased by a factor of ten. Such high sampling intervals are unsuitable for real-time control.

IDENTIFICATION OF THE PUMA 560 PARAMETERS USING THE DSP BASED HARDWARE

The dynamic control of an industrial manipulator involves the determination of the inputs (torques or voltages) for the actuators which operate at the joints so that a set of desired values for the positions and velocities for the manipulator is achieved. Virtually all forms of dynamic control involve the use of a system model for the design of control algorithms. In the case of adaptive/self tuning control, the model used is generally a discretized one which takes the form of a time series model containing any linear and nonlinear terms which might be present in the system.

A General Time Series Model of the PUMA 560

The time series model for each joint has the form:

$$\begin{aligned} y(kT) = \; & A_0 + A_1 y[(k-1)T] \\ & + A_2 y[(k-2)T] \ldots\ldots + B_1 u[(k-1)T] \\ & + B_2 u[(k-2)T] \ldots\ldots + f[kT] + M(kT) \end{aligned}$$

where u(kT) is the model input, or joint voltage, and y(kT) is the output or joint position at time kT. A_i and B_i are coefficients of the linear portion of the model, $f(.)$ is the discretized joint nonlinearities contained in the torque terms of the robot model and M(.) represents modelling errors.

An autoregressive moving averages (ARMA) model can then be assumed for each joint based on this time series model. It takes the form of the following difference equation

$$y(k) = A(q^{-1})y(k) + B(q^{-1})u(k) + h(k) + e(k)$$

where k refers to the sampling interval of the discretization. The term h(k) represents a forcing term including the nonlinear effects present in the torque terms. In the equation

the error e(k) represents a zero mean white noise. $A(q^{-1})$ and $B(q^{-1})$ are polynomials with q^{-1} being the backward shift operator.

Tests Performed

The first three joints of the PUMA were put through a series of tests, the results of which were used for the identification of the model parameters. Joints one to three were put through their full joint range at two different speeds.

1) slow trajectory unloaded,(VAL speed 50) and
2) fast trajectory unloaded,(maximun VAL speed 100).

The parameters of the time series model were estimated from the input/output pairs using four different on-line estimation methods.

Method 1: Recursive Least Squares (RLS)

By assuming the coupling terms are small and that the PUMA 560 system parameters are slowly time-varying [11] with negligible measurement noise, it is possible to apply the simplest form of RLS to the identification of this robot's parameters. This model can be written as:

$$y(k) = A(q^{-1})y(k) + B(q^{-1})u(k) + e(k)$$

If the parameter vector Θ and the regressor information vector Φ are defined as

$$\Theta^T = (a_1,...,a_n;b_1,...,b_n) \qquad \text{and}$$

$$\Phi^T = [\ y(k-1),...,y(k-n);\ u(k),...,u(k-n+1)]$$

the model can then be written as:

$$y(k) = \Theta^T.\Phi(k-1) + e(k)$$

The parameter estimation problem is to find the estimates of the unknown parameters which minimize the loss function:

$$E(\Theta_i) = \frac{1}{m+1} \sum_{i=1}^{m} [\,e_i(k)\,]^2$$

where $e_i(t)$ is the prediction error in the parameters of joint i and m is the number of parameters being estimated. The principle underlying least squares is that by minimizing the prediction error it is possible to minimize what is unexplained in the model. The solution to the Least Squares problem is furnished by the following recursive equations:

$$\Theta_i(k) = \Theta_i(k-1) + P(k)\Phi(k-1) \cdot [\ y_i(k) - \Theta^T{}_i(k-1)\Phi(k-1)]$$

$$P(k) = \frac{1}{\mu}\left[P(k-1) - \frac{P(k-1)\Phi(k-1)\Phi^T(k-1)P(k-1)}{\mu + \Phi^T(k-1)P(k-1)\Phi(k-1)} \right]$$

where P is the covarience matrix (2nx2n) of the estimation errors and where μ is what is known as the forgetting factor. The P matrix is the positive definite measure of the estimation error and its elements tend to decrease as the estimates converge to their true value. It is therefore necessary to initialize the elements of this matrix to some large

value when the initial estimates are poor. The *forgetting factor* μ is set to a value less than unity to ensure the estimation procedure continues to track parameter variations (i.e. the procedure does not *fall asleep*).

Method 2: Modified RLS

This method of is based on the least squares model just described. This more comprehensive autoregressive model can be written as:

$$y(k) = A(q^{-1})y(k) + B(q^{-1})u(k) + h + e(k)$$

where h is a forcing term intended to include the nonlinear effects of torque-dependent terms.

In this case, the parameter estimates and the regressors can be written in the following vector format:

$$\Theta^T = (a_1,...,a_n;b_1,...,b_n;h_1) \qquad \text{and,}$$

$$\Phi^T = [\ y(k-1),...,y(k-n);\ u(k),...,u(k-n+1);1]$$

The autoregressive model can be again written as:

$$y(k) = \Theta^T.\Phi(k-1) + e(t)$$

This is the format required to apply the loss function equation for the minimization of the prediction error.

Method 3: Extended Least Squares (ELS)

This method attempts to estimate a model for the noise present in the system, as well as the system model itself. This model can be written in time series form as follows:

$$y(k) = A(q^{-1})y(k) + B(q^{-1})u(k) + C(q^{-1})e(k) + d(k)$$

where $C(q^{-1})$is the polynomial containing the parameters of the noise model and d(k) is called the loaded disturbance variable.

In this case, the parameter estimates and the regressors can be written in the following vector format:

$$\Theta^T = (a_1,...,a_n;b_1,...,b_n;c_1,..,c_n) \qquad \text{and,}$$

$$\Phi^T = [\ y(k-1),...,y(k-n);\ u(k),...,u(k-n+1);$$
$$e(k),...,e(k-n)]$$

The autoregressive model can be written as:

$$y(k) = \Theta^T.\Phi(k-1) + e(t).$$

A second order model structure for both the noise and the system model itself means a total of six parameters have to be estimated.

Method 4: Nonlinear Extended Least Squares (NELS)

This method attempts to estimate a model for the residual as a combination of linear and nonlinear functions. This model can be written as follows:

$$y(k) = A(q^{-1})y(k) + B(q^{-1})u(k) + C(q^{-1})e(k) + N(k)$$

where $C(q^{-1})$is the polynomial containing the parameters of the noise model and N(k) is a nonlinear polynomial defined by:

$$N(k) = n_1u^2(k) + n_2u^3(k)$$

In this case, the parameter estimates and the regressors can be written in the following vector format:

$$\Theta^T = (a_1,...,a_n;b_1,...,b_n;c_1,..,c_n;n_1,n_2) \quad \text{and,}$$

$$\Phi^T = [\ y(k-1),...,y(k-n);\ u(k),...,u(k-n+1);\ e(k),...,e(k-n);u^2(k),u^3(k)]$$

The autoregressive model can be again written as:

$$y(k) = \Theta^T.\Phi(k-1) + e(t)$$

A second order model structure for the system model, the noise model and the nonlinearity means that a total of eight parameters have to be estimated.

Identification Results

The RLS method is unsuitable for identification of the robot parameters. The MRLS method models the robot more accurately but fails to show any substantial improvement in convergence time without good initial estimates. The ELS method is more accurate than the previous methods and shows rapid convergence even without good initial estimates. The method of NELS was found to model the robot most accurately and shows similar convergence properties to the method of ELS.

The graphs above(Fig.1 to Fig.4) show the magnitude of the loss function versus time. These prove conclusively that the NELS method, with the smallest loss function is the most accurate method for identification of the robot parameters.

CONCLUSION

This paper outlines the motivation for the development of a more flexible control structure and discusses the capabilities of this DSP based control scheme. Identification results presented show one of the controller facilities.

Future work on this controller involves the implementation and comparision of several robotic control algorithms. These algorithms will range from the very simple (i.e. fixed gain PD control) to the very complex (i.e. computed torque , non-linear self-tuning control). The hardware developed is of sufficient speed to implement these complex controllers with an adequate sampling period.

REFERENCES

[1] Mclidy A. & Goldenberg A.A., "Operation of the PUMA 560 Without VAL", Robotics Proc. Robots 9, 1985.

[2] Unimation (Europe) Ltd., "PUMA 560 Mk2 Robot System Technical Manual", Sept. 1985.

[3] Lee C.S.G. et al, "Hierarchical Control Structure Using Special Purpose Processors for the Control of Robot Arms", Proc. of the IEEE Pattern Recognition and Image Processing Conference", pp 634-640, 1982.

[4] Ringwood J.V., "Control Strategies for Robotic Manipulators", Proc. IMC-6 Conference on Advanced Manufacturing Technology, Dublin City University, Sept 1989.

[5] Guruasavaraj K.H., "Implementation of a Self-Tuning Controller Using Digital Signal Processing Chips", IEEE Control Systems Magazine, June 1989.

[6] NEC,"μPD77230 Digital Signal Processor Product Description", 1989.

[7] Khosla P.K. & Kanada T., "Experimental Evaluation of Feedforward Compensation and Computed-Torque Control Schemes", Proc. of the American Control Conf., Seatle, WA, June 1987.

[8] Astrom K.J., "LQG Self-Tuners", IFAC Symposium on Adaptive Systems, San Francisco, 1983.

[9] Grimble M., "Implicit and Explicit LQG Self Tuning Controllers", Automatica, Vol 20, No. 5. 1984.

[10] Lelic M. & Wellstand, "A generalized Pole Placement Self-Tuning Controller - An Application to Manipulator Control", Control Systems Centre Report No. 658, UMIST, Manchester, August 1986.

[11] Koivo A.J.& Guo H., "Adaptive Linear Controller for Robotic Manipulators", IEEE Trans. on Automatic Control, Vol AC-28, No. 2, Feb 1983.

DIAGRAMS

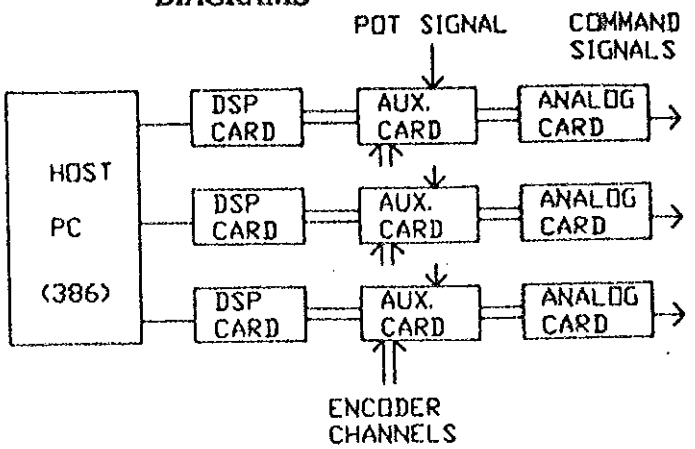

Fig.1 System Block Diagram

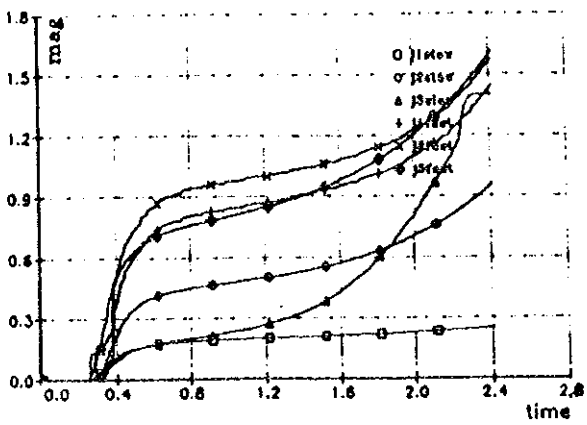

FIG.2 RLS LOSS FUNCTIONS (FAST AND SLOW).

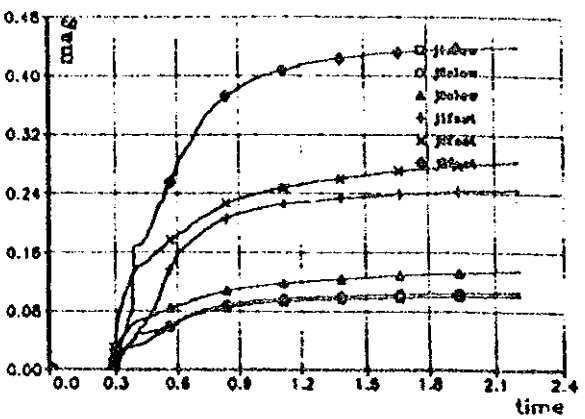

FIG.3 MRLS LOSS FUNCTIONS (FAST AND SLOW)

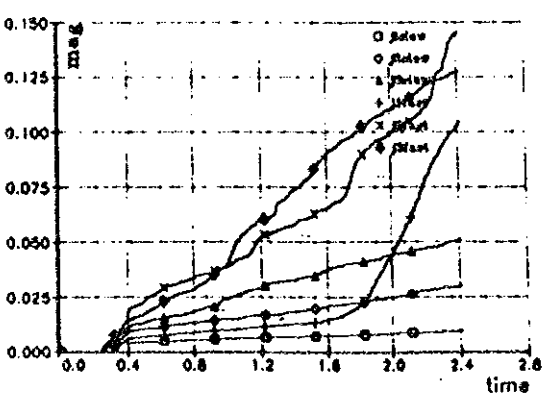

FIG.4 ELS LOSS FUNCTIONS (FAST AND SLOW)

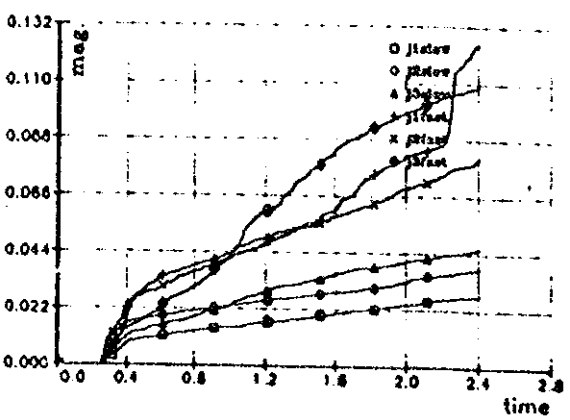

FIG.5 NELS LOSS FUNCTIONS (FAST AND SLOW)