Workstation for integrated system design and development

K. Cosic, I. Kopriva and I. Miler

SIMULATION 1992; 58; 152
DOI: 10.1177/003754979205800305

TECHNICAL ARTICLE

SIMULATION 58:3, 152-162

ISSN 0037-5497/92 $3.00 + .10
Printed in the United States of America

Workstation for integrated system design and development

K. Cosic, I. Kopriva, I. Miler
Faculty of Electrical Engineering
University of Zagreb
41000 Zagreb, Unska 3
Croatia, Yugoslovia

Hardware in the loop simulation has been proven as a very cost effective method in design, development, modification and testing of sophisticated industrial control systems. The Workstation for Integrated System Design and Development presented in this article allows: highly efficient real time simulation of dynamic systems using the available multiprocessor resources, microprocessor control system design and implementation using suitable software package for synthesis and code generation, testing and verification of microprocessor based controller using real time interaction with multiprocessor simulator. The hardware configuration of Workstation for ISDD is based on inexpensive general purpose single board computers, which are very attractive and more favorable in speed/cost ratio in relation to the other powerful single processor solution. The appropriate examples show the applicability of these multi-microprocessor resources in: real time simulation of linear time invariant systems, comparative analysis of different discretization techniques and digital control system design and testing.

Key words: multiprocessor resources, digital real time simulation, code generation, ordinary differential equations, system decomposition, numerical integration, discretization methods, integrated system design and development, task allocation.

Introduction

The software tools and hardware resources which constitute the Workstation for Integrated System Design and Development (ISDD) provide:

- highly efficient real time simulation of dynamic systems using the available multiprocessor resources,
- microprocessor control system design, development and implementation using suitable software package for synthesis and code generation,
- testing, evaluation, validation and verification of microprocessor based controllers using real time interaction with multiprocessor simulator.

The system which provides abilities for real time simulation of plants with hardware components integrated in the simulation loop, would be very helpful in the process of integration of complex industrial control systems. Testing of embedded microprocessor control hardware and testing whether its software operates correctly in a nearly realistic environment, can be performed using the Workstation for Integrated System Design and Development. Therefore, such testbed platform may be used as general purpose Control System Test Equipment for evaluation of industrial controllers. By using such resources and concepts it is possible to predict accurately real world performance and behaviour of closed loop system , to explore more design options, and to solve problems that couldn't be solved with traditional approach based on non-real time simulation. Such simulation technology, therefore, has been proven as a very cost effective method which reduces time and money necessary for

design, development, modification and testing of sophisticated control systems without risks and dangers [1; 14; 17; 20, etc.].

Real time simulation using multi-microprocessor resources

The real time simulations using standard microprocessors are very attractive due to price/performance ratio. Therefore, a number of attempts to interconnect relatively inexpensive micros have been made over the past years such as [2; 19], etc. This approach was a time consuming job which required extensive research and experimentations not directly related to the application objectives.

The concept which has been applied in this article involves several phases. The first step requires suitable decomposition of mathematical model given by algebraic and ordinary differential equations. Since these equations are defined over continuous time domain some kind of discretization or numerical integration techniques must be employed. Finally, it is necessary to perform code generation for discretized modules and to perform mapping of discretized mathematical modules onto parallel architecture resources.

System decomposition

The first step in this procedure requires decomposition of a mathematical model of a given dynamic system into a number of hierarchical functional modules or blocks of different complexity. Such decomposition of a complex system into multiple low level computational modules enables adaptation of numerical methods and period of discretization to each module.This structural and dynamic decomposition, which corresponds to physical topology of the system, leads to the highly efficient digital real time simulation. A heuristic method for system decomposition is based on the physical topology of the system. The main advantage of this approach is that the processing modules are associated with physical sections, and therefore modules implementation and verification can be easily done. In addition, variables used in the interprocessor communication have physical meaning, which can aid in understanding and interpretation of the model behaviour. Furthermore, by exchanging only the output variables between blocks it is possible to minimize interprocessor communication requirements. Program design and coding is simplified, while checking, testing and debugging are also easier.

Numerical integration

The second step related to the numerical discretization/ integration transforms original continuous mathematical model into equivalent discrete one. Discretized mathematical model must guarantee desired level of approximation with minimal arithmetic complexity. Assignment of suitable integration or discretization method, and suitable period of discretization to each module or block, requires extensive non-real time analysis, checking, testing and verification.

The choice of the optimal integration methods i.e. tuning the method to each separate module requires full understanding of the character and dynamics of each module and properties of numerical methods suitable for real time simulation [10; 11]. The separation of differential equations according to their types and dynamics i.e. separation linear/linearized differential equations from nonlinear, fast portions of the problem from slow portions etc.,has great influence on the efficiency and the speed of simulation [18]. For complex nonlinear modules or blocks, which don't allow further physical decomposition, and single processor real time implementation, it is necessary to use some technique for equation segmentation or numerical methods for parallel integration [7].

Code generation

It is well known that standard powerful simulation languages like CSSL, ACSL, Matrix etc., are not suitable for real time simulation due to their massive amount of code. On the other hand, assembly coding significantly depends on a programmer's experience and skills and is formidable, time consuming and error prone. In order to enhance productivity of the process of developing real time simulation software, i.e. to reduce model time development and improve quality and reliability of the software products, different automatic tools and techniques have been applied [9; 12; 22]. Taking into account that the intensity of numerical computations depends on complexity of modules, applied numerical methods, frame time and certainly on the efficiency of code used for implementation, the code generators adapted to different type of modules and different discretization/integration techniques have been used. It means that software tools for code generation contain code generators adapted to different type of modules, numerical methods and techniques for their solutions [6]. In order to achieve the speed of execution, code portability and flexibility realized code generators can produce assembly 86/87 source code for linear modules or hardware independent C – code for linear and nonlinear modules.

Task allocation

The last step requires distribution of derived discretized modules onto available multiprocessor

resources. There are two different options in task allocation strategy which depend on application requirements [4].

In the first case, architecture of the peripheral multiprocessor system is completely defined and determined by the type and number of available processors, their performance and way of their communication. The computing tasks involved in solution of simulation problem, in this case, must be partitioned on the available processors in order to minimize idle time of each processor, i.e. to attain equal load balancing. In the second case the architecture of multiprocessor system is not given and has to be determined by this procedure. In this case task allocation strategy must provide minimization of hardware requirements necessary for simulator realization.This is essential in those applications where simulator configuration must be duplicated in large number of copies.

The input to the task partitioning procedure is Module Information File which contains all necessary information about modules interconnections, discretization/integration method, implementation times for each module, predecessor and successor modules and types of input and output signals for each module. On the basis of this information task partitioning algorithm groups modules in a cluster, taking care of real time constraints and limitations for each separate module.The cluster represents a set of modules which are assigned to the same microprocessor.

The number of software modules which will be assigned to one microprocessor depends on arithmetic complexity of discretized modules and real time constraints. Possible assignment can be several modules to one microprocessor, or several microprocessors to one module. This relationship depends predominantly on the processing power of each processing element, and complexity of discretized modules and their dynamics.

Workstation for Integrated System Design and Development

The Workstation for Integrated System Design and Development has been designed and realized using suitable multi microprocessor resources, Figure 1. This

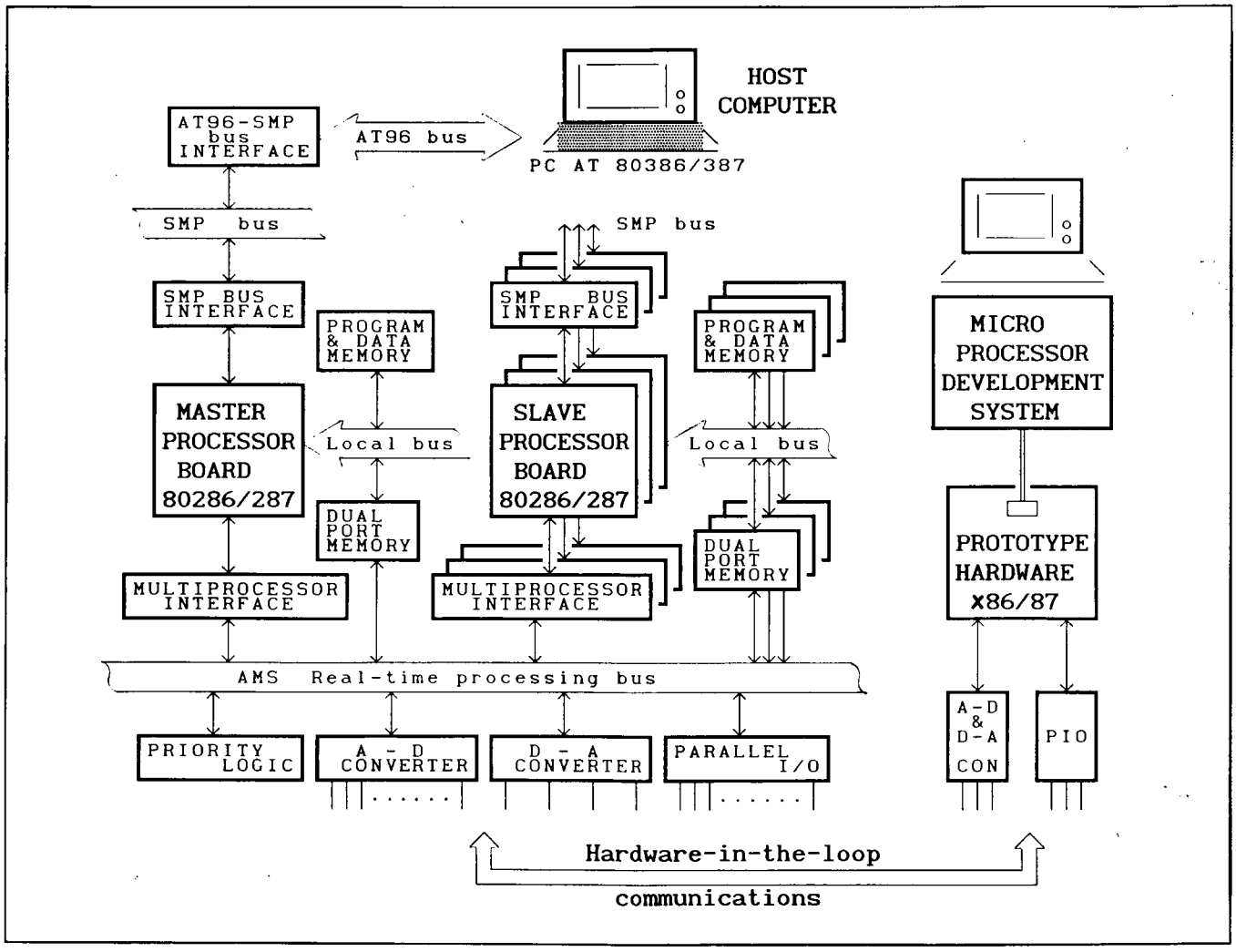


Figure 1. The hardware configuration of the multiprocessor simulator for testing of prototype hardware

configuration is based on inexpensive general purpose single board computers, which are attractive and more favorable in speed/cost ratio in relation to the other single processor solution.

Hardware configuration is based on PC 386/387 as the host and the attached peripheral homogeneous tightly coupled multiprocessor system based on Siemens single board computers AMS–M26–A81 with 80286/287 micro's [4]. Five single board computers were used to realize simulator processing hardware. Interprocessor communication is realized over 16-bit data bus AMS–M using dual port memories and suitable arbitration logic. Access to the input of external analog signals has been provided through analog to digital input board AMS–230–A1 with 12 bits and 16/32 channels. Access to the analog output world is realized by AMS–M596 customer oriented interface board with four digital to analog converters. Modular design of this configuration presented by Figure 2 provides a number of benefits concerning its flexibility, reconfiguration, application, modification and maintenance.

Furthermore, microprocessor development system Tektronix 8540 included in this configuration enables efficient development, testing and verification of custom design hardware and software through the interaction with the simulator multiprocessor system. It means that such organization of hardware resources offers various experimental abilities in real time simulation and its application in design, testing and modification of microprocessors control system, in operators training and digital control system education.

In order to illustrate the abilities of software tools integrated in the Workstation for ISDD, a few examples will be presented in this article. They show simplicity and efficiency of real time simulation of LTI modules, methodology for optimization of discrete time model for real time simulation using suitable compensation of frequency distortions, multiprocessor comparative real time analysis of different discretization techniques and abilities of this system for control system design, implementation, evaluation and real time simulator based testing.

Real time simulation of linear time invariant modules

This example illustrates simplicity of procedure for real time simulation of linear time invariant modules using available software resources for automatic: discretization, assembly source code generation, coefficient scaling, configuration of I/O macros (a/d, d/a, real time clock), determination of speed up factor, module assignment, assembling, linking, downloading to selected microprocessor and finally performing real time simulation.Complete procedures are fully automatized and require only few seconds from user specification of input model to d/a output which can be monitored on oscilloscope, Figure 3.

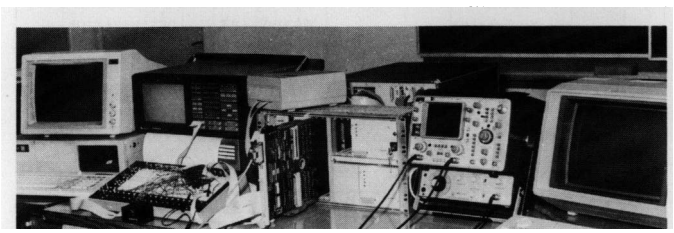


Figure 2. The experimental laboratory version of the multiprocessor simulator system

The process of real time simulation of linear/linearized module will be illustrated by the next example which requires following user specifications:

- Transfer function or state space form of mathematical model.
- Required accuracy of discrete equivalent model.
- Type of arithmetic used for implementation, i.e. fixed point or floating point.
- Selection of discretization procedure among many different discretization techniques available.
- I/O information concerning the automatic configuration of I/O macro's for mux, a/d and d/a channels and real time clock.

The mathematical model of linear module is given by

$$G(s) = \frac{0.01856}{5.22824\,10^{-3}\,s^2 + 0.0182935\,s + 1} \qquad (1)$$

Using the modified step invariant method we obtain

$$x(k+1) = \begin{bmatrix} 0.90272 & 2.99158E{-}2 \\ -5.7241 & 0.79804 \end{bmatrix} x(k) + \begin{bmatrix} 3.51897E{-}3 \\ 9.26902E{-}3 \end{bmatrix} u(k) \qquad (2)$$

and maximum possible frame time T = 32.414 [ms] according to the specified accuracy requirements [5].

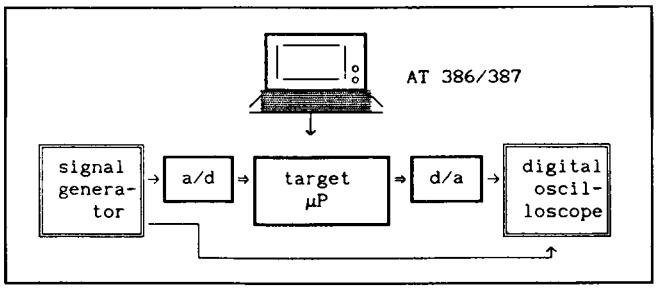


Figure 3. The hardware configuration for real time simulation of single LTI module

After the specification of I/O requirements, corresponding I/O macro's will be selected and assembly source code for discretized model (2) will be generated. Single step execution of produced binary code on peripheral host microprocessor has objectives to measure the implementation time of this module on single processor. After that, a user receives information about speed up factor K, i.e. ratio between frame time and implementation time, and suggestion for its allocation. If K is greater than one, task allocation scheduler will assign this module to one μP. Otherwise, it will perform suitable partitioning of discrete module which will guarantee real time simulation. Optionally, a user can specify its own task allocation strategy which will be checked in order to provide real time execution. The real time response of specified LTI module for selected input is shown by Figure 4.

Optimization of discrete time model for real time simulation

The real time simulations of dynamic systems are constant challenge for synthesis of new numerical methods which lead to the minimization of arithmetic complexity and which are robust enough, i.e. retain high level of accuracy with relatively high integration step. Using such techniques it is possible to reduce, tremendously, arithmetic complexity of given problem. Since, complex dynamic systems contain many isolated linear or linearized modules, their highly efficient discretization is of special interest.

For an original continuous system defined by

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{3}$$

$$y(t) = Cx(t)$$

or corresponding transfer function G(s), it is necessary to design equivalent discrete model

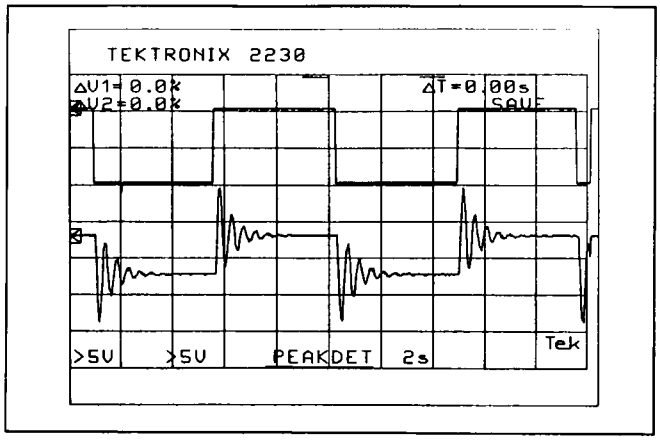


Figure 4. The real time response of specified transfer function

$$x(k+1) = \Phi x(k) + \Gamma u(k) \tag{4}$$
$$y(k) = Cx(k)$$

Discrete equivalent must satisfy accuracy criteria defined in time or frequency domain with maximal value of speed up factor

$$K = T / T_i \tag{5}$$

where T denotes discretization period and T_i implementation period.

The basic concept of discretization procedure which enables optimization of equivalent discrete time model is shown by Figure 5 [21].

State space formulation of the system defined by Figure 5 is given by [3; 5].

$$x[(k+1)T] = e^{AT}x(kT) + \Gamma(\lambda_0, \gamma_0, T)u(kT)$$

$$y(kT) = Cx(kT) \tag{6}$$

where are

$$\Gamma(\lambda_0, \gamma_0, T) = \lambda_0 [A^{-1}(e^{AT} - I) + \gamma_0 T(e^{AT} - I)]B =$$

$$= \lambda_0 [T\psi_0 + \gamma_0 T^2 \psi_0]B$$

and

$$\psi_0 = \sum_{j=0}^{\infty} \frac{A^j T^j}{(j+1)!} \tag{7}$$

The optimal values of compensation parameters λ and γ can be obtained using nonlinear programming techniques according to Figure 6 [5; 15].

The objective function for minimization of differences between frequency characteristics of continuous and equivalent discrete system is given by

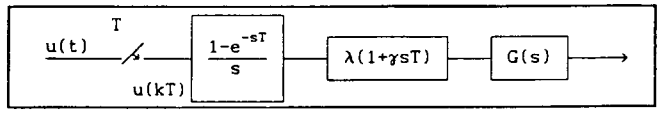


Figure 5. The concept of compensated step invariant method

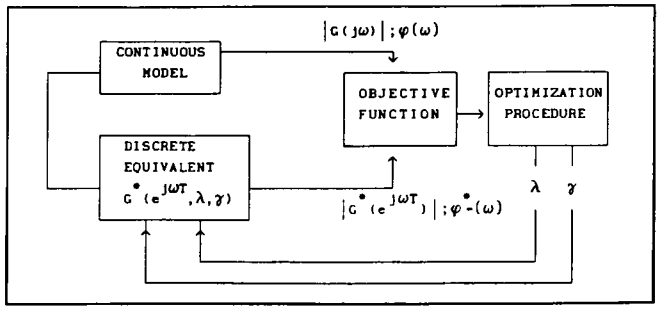


Figure 6. The procedure for tuning of compensation filter parameters λ and γ

$$IP(\lambda,\gamma) = \sum_{i=1}^{N} \{\mu_1(\omega_i) * |20\log|G(j\omega_i)| - 20\log|G(e^{j\omega_i T})| | + \mu_2(\omega_i) * * |\varphi(\omega_i) - \varphi(e^{j\omega_i T})| \} \quad (8)$$

where weighting functions $\mu_1(\omega_i)$ and $\mu_2(\omega_i)$ are defined by

$$\mu_1(\omega_i) = \mu_2(\omega_i) = |G(j\omega_i)| \quad (9)$$

The results of the optimization are shown by Figure 7.

The optimal values of compensation filter parameters in relation to the starting point $\lambda_s = 1$ and $\gamma_s = 0$ are determined by

start:	$IP_s = 1$	$\lambda_s = 1$	$\gamma_s = 0$
optimum:	$IP_o = 0.0216273$	$\lambda_0 = 0.98$	$\gamma_0 = 0.5$

It means that distortions in frequency domain introduced by standard step invariant method can be reduced $IP_s/IP_0 = 46.2378$ times after the process of optimization.

Comparative analysis of different discretization techniques

The frequency distortions introduced in the closed loop system during the process of discretization must be controlled since they can have significant influence on the simulator usefulness for hardware in the loop testing or man in the loop training. Many available discretization techniques are integrated in the ISDD system. Using available multiprocessor resource it is possible to perform their comparative analysis in time and frequency domain and to estimate their real time performance. Measurement of computation time necessary for module implementation enables computation of the speed up factor for each method and their comparison, since the criteria function is same for all different techniques.

The hardware configuration used for experimentation in this example is shown in Figure 8.

The objective of this experiment is to synthesize and to realize discrete equivalent of original continuous system which would approximate quasi reference solution with predefined accuracy requirements and maximum speed up factor.

Comparative analysis of different discretization techniques for second order system (1) is given by the Table 1.

Table 1 contains maximum possible values of discretization period T obtained by different discretization techniques for the same accuracy condition. Accuracy requirement is prescribed in time domain and is equal to 3% for common composite input signal which disturbs tested systems in the whole bandwidth range.

Highly efficient discrete time model is produced by compensated step invariant method. Its testing has been performed according to Figure 9.

The comparative analysis of real time responses for common frame time equal to T = 32.414ms is shown by Figure 10.

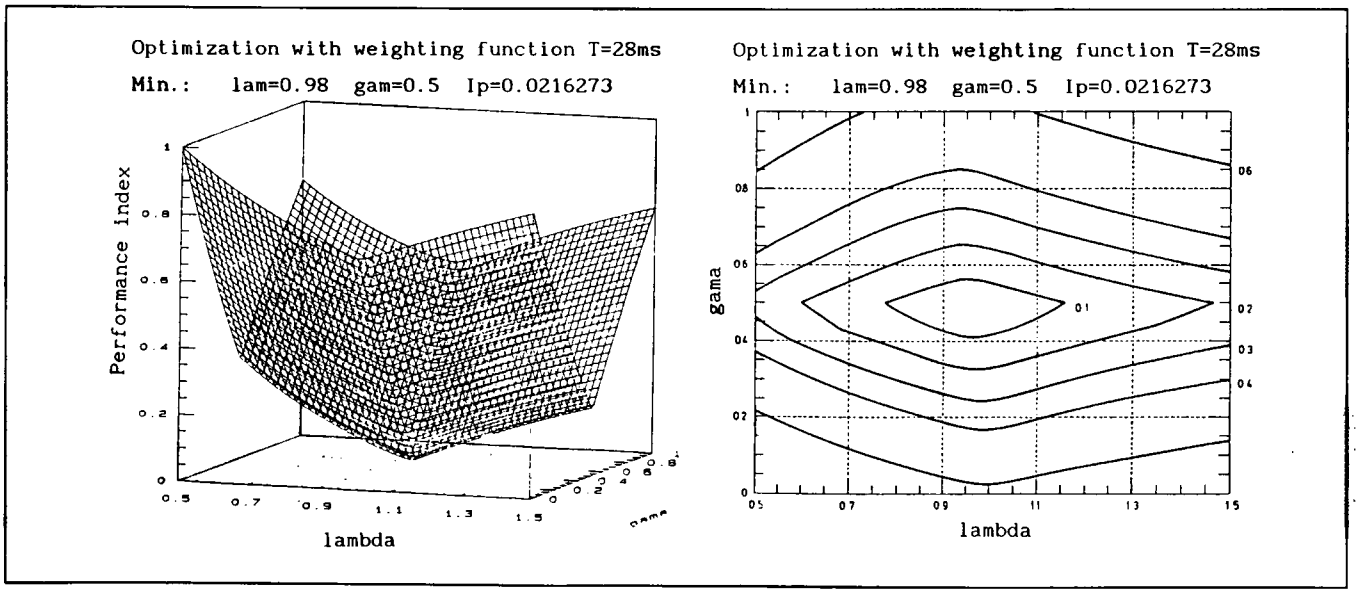


Figure 7. The tuning of compensation filter parameters

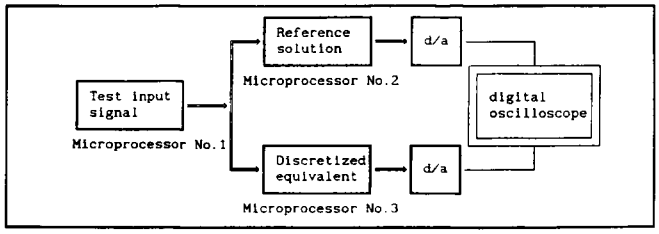


Figure 8. The hardware configuration for testing different discretization techniques

Table 1. Performance analysis of different discretization techniques for the same accuracy requirements

discretization method	standard step invariant	modified step invariant			
		$\mu(\omega)=1$	$\mu(\omega)=\lvert G(j\omega)\rvert$	$\mu(\omega)=\lvert F(u(t))\rvert$	$\lambda=1$ $\gamma=0.5$
maximal T [ms]	8.486	18.726	32.414	28.832	32.91
speed up factor K	26.55	58.59	101.42	90.28	102.97

Euler explicit	Euler implicit	bilin. trans.	AB-2	standard ramp inv.
1.582	1.764	23.334	10.78	23.429
7.15	5.52	52.7	46.1	52.91

Discretization of analog controller, its implementation and simulator based testing

In discretization process of analog controller and its microprocessor implementation the main objective is to retain the performance and specifications of the original analog closed loop system. It is well known that the behaviour of redesigned analog control system predominantly depends on applied discretization techniques and choices of discretization period. The following example shows that by using compensated standard step invariant method, properties of original continuous system can be retained even with a higher discretization period.

Hardware configuration for testing the influences of discretized analog controller on the closed loop behaviour, using real time simulator of the plant, is shown in Figure 11.

In order to demonstrate the abilities of available software tools for discretization and code generation, and abilities of available multiprocessor simulator for its testing, verification and evaluation, let us consider the following example [13].

$$D(s_i)=\frac{3.8\left(1+0.034013s\right)}{\left(1.156925\bullet10^{-5}s^2+6.802721\bullet10^{-3}s+1\right)} \qquad (10)$$

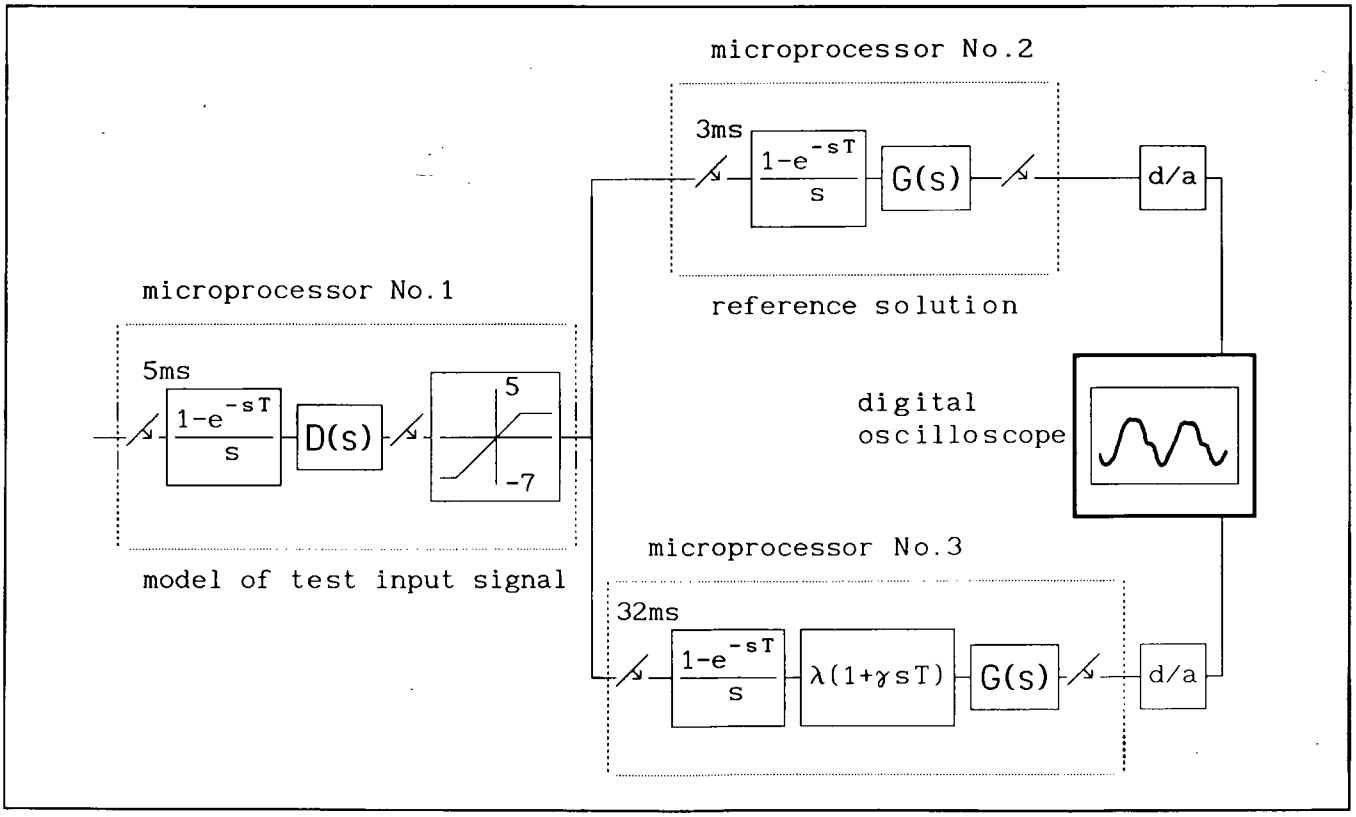


Figure 9. Multiprocessor configuration for testing of compensated step invariant method

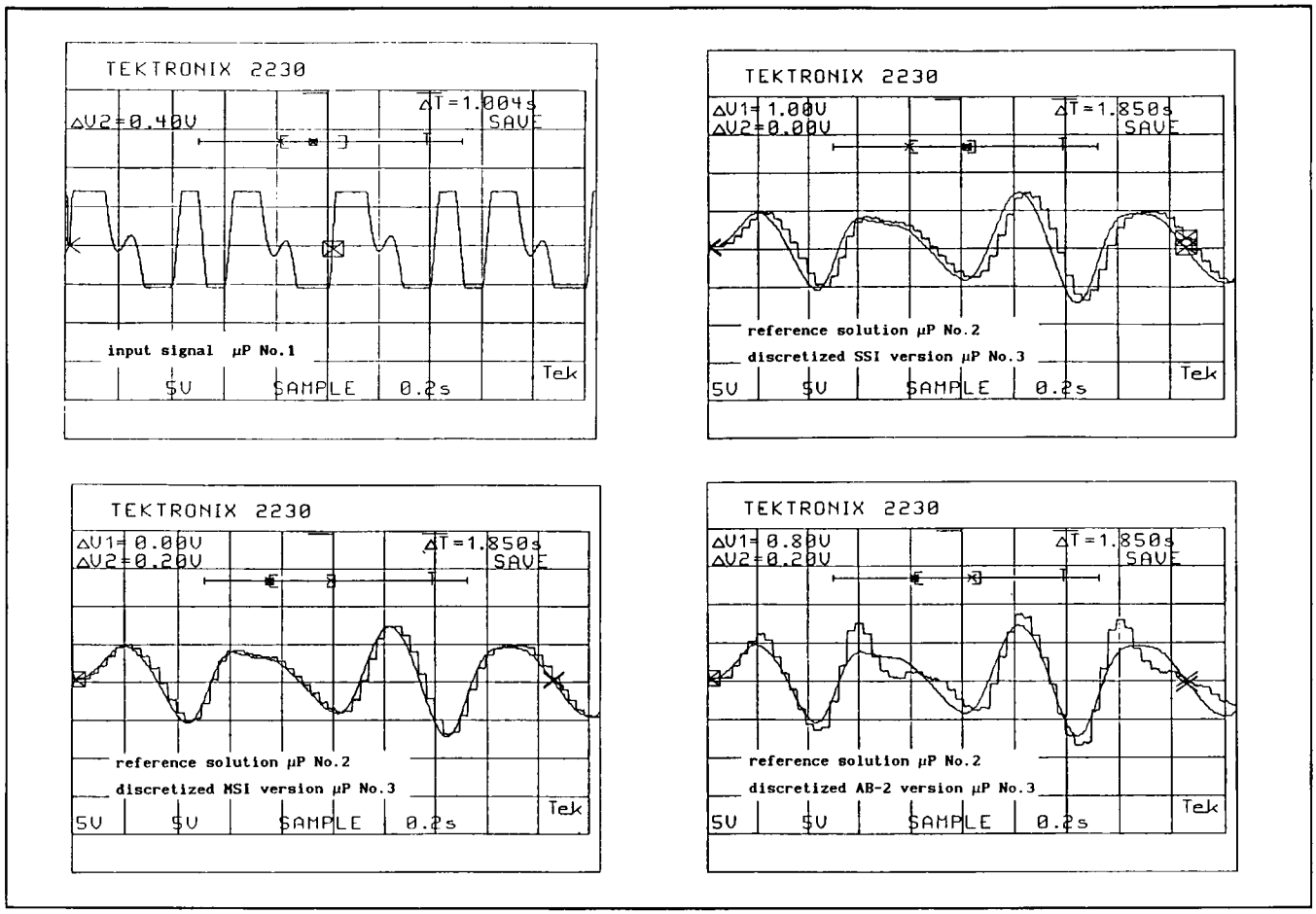


Figure 10. Real time responses of SSI, MSI and AB–2 method for the same discretization period and same input signal

$$G(s) = \frac{234.741784}{(s^2)} \qquad (11)$$

Discretization technique of analog controller based on the compensated step invariant method is shown by Figure 12.

This experiment requires:

- plant discretization, its code generation and task allocation to the selected microprocessor,
- controller discretization, its code generation and task allocation to target microprocessor.

Real time multi-rate simulation of closed loop system synthesized according to Figure 12., for T_2 = 1ms and T_1 = 1,2,3,4,5ms is shown by Figure 13.

Discretization technique of analog controller based on standard step invariant method is shown by Figure 14.

Time responses of real time multi-rate simulation using T_2 = 1ms and T_1 = 1,2,3,4,5ms are shown by Figure 15.

By comparative analysis of responses obtained on Figure 13 and 15 it is easy to notice significant advantage of compensated, i.e. modified, step invariant method in application concerning discretization technique of analog controller.

The pretested controllers can be removed from the simulator testbed and embedded in the real plant environment using only new I/O configuration.

Conclusion

This paper has presented a cost–effective approach to digital real time simulation of dynamic systems. The hardware and software environment of workstation for integrated system design and development has been described. By appropriated examples we have shown the applicability of multi–microprocessor resources in: real time simulation of linear time invariant systems, comparative analysis of different discretization techniques and digital control system design and testing. Special attention has been paid to the new method for discretizing linear time invariant systems based on optimization in frequency domain by using nonlinear programming technique. The proposed physically based partitioning approach is highly flexible and

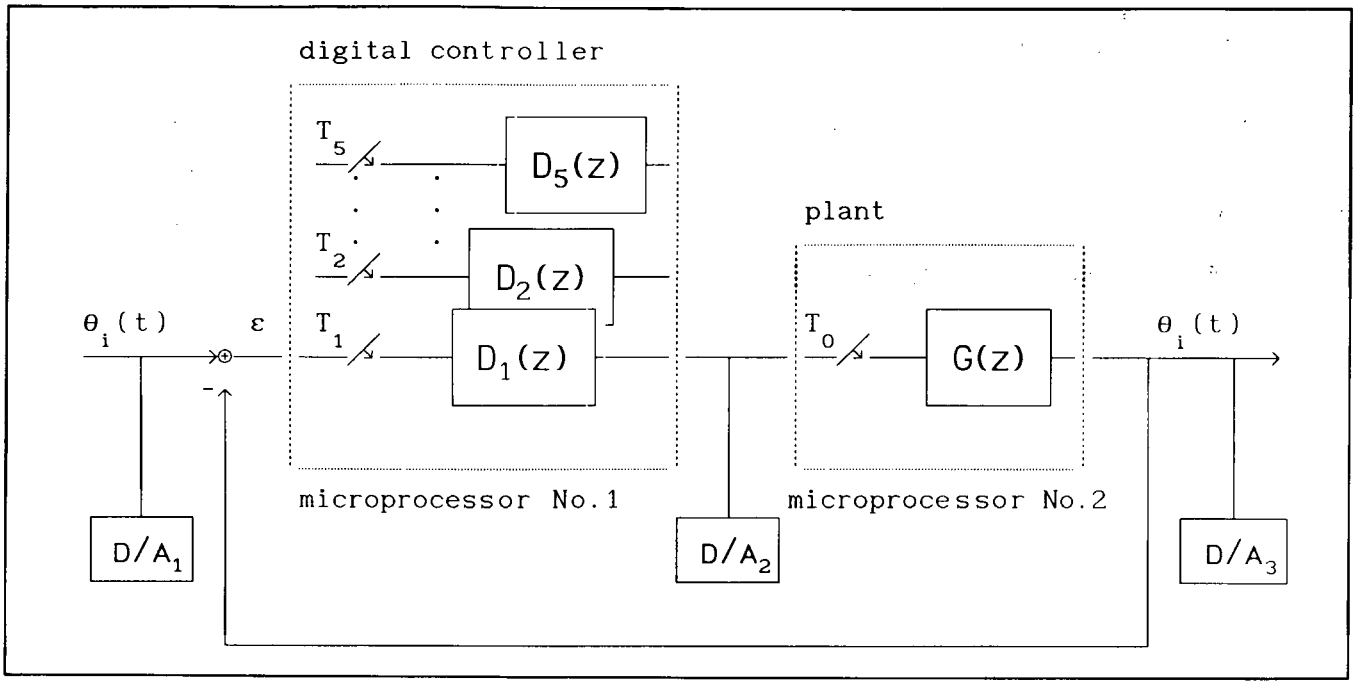


Figure 11. The hardware configuration for real time testing of discretized analog controller

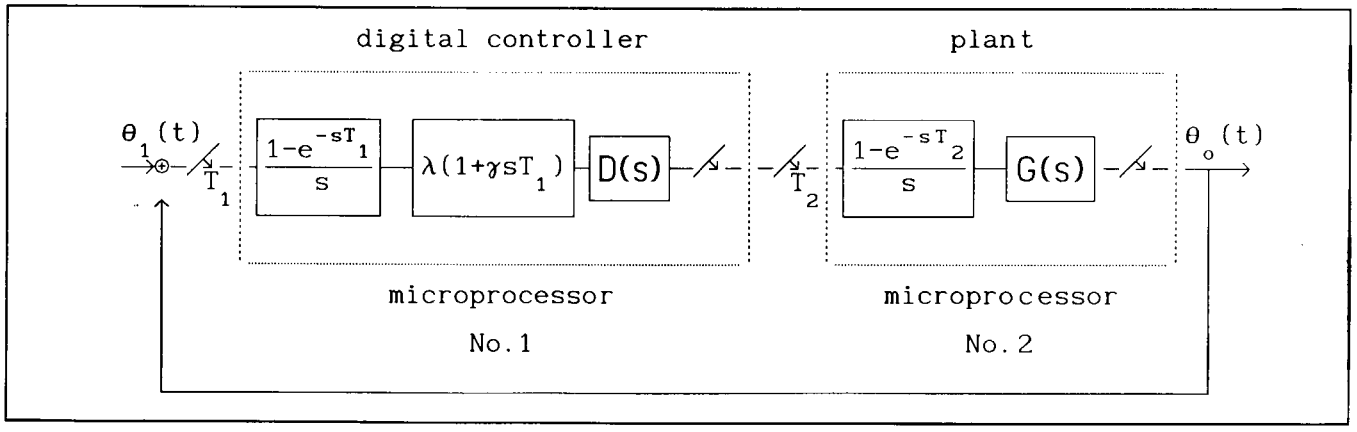


Figure 12. Discrete equivalent of closed loop system based on modified step invariant method for discretization of analog controller

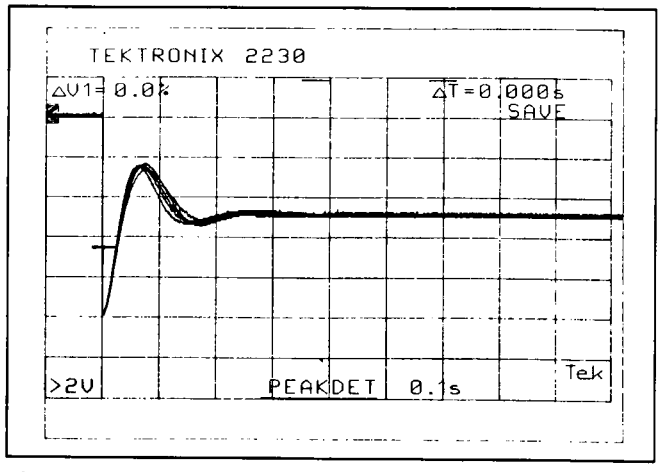


Figure 13. Multi-rate multiprocessor real time simulation of closed loop servo system using MSI discretization technique

promising. By using more powerful peripheral array processors like transputer networks, digital real time simulation of more complex dynamic systems can be achieved.

References

1. Berthoumieux, J.M., La Cava, A.I. "Real time Computer Dynamic Simulation in Chemical Process Control Education", *Proceedings of the Summer Computer Simulation Conference*, 1985.
2. Blech, R.A., Arpasi, D.J. "Hardware for a real-time multiprocessor simulator", *Proceedings of the Summer Computer Simulation Conference*, 1985.
3. Cosic K.: "Design and implementation of discrete time model for real time digital simulation", *All About Simulators*, Simulator series, Vol. 14, No.1, 1984.

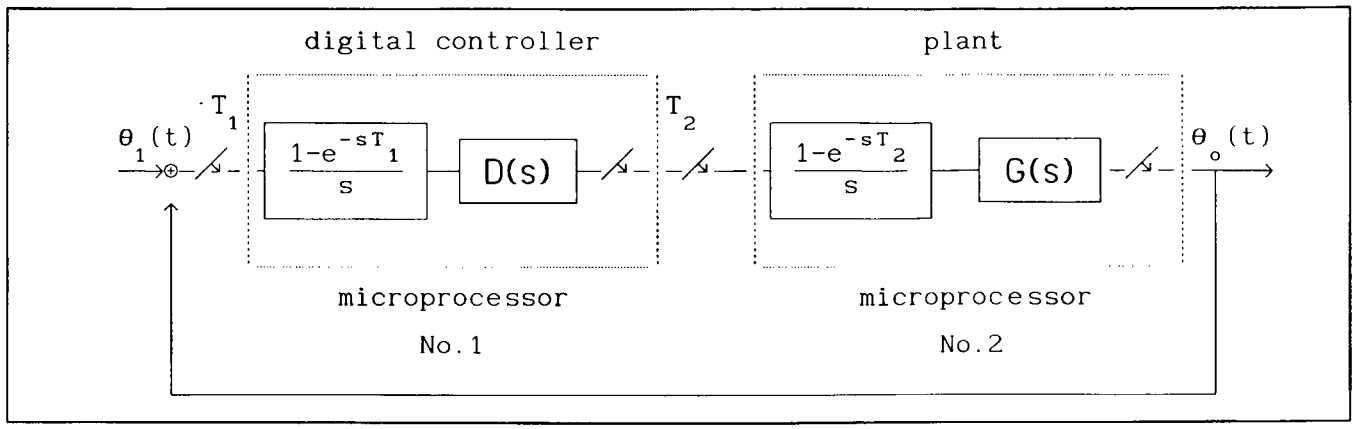


Figure 14. Discrete equivalent of closed loop system based on standard step invariant method for discretization of analog controller

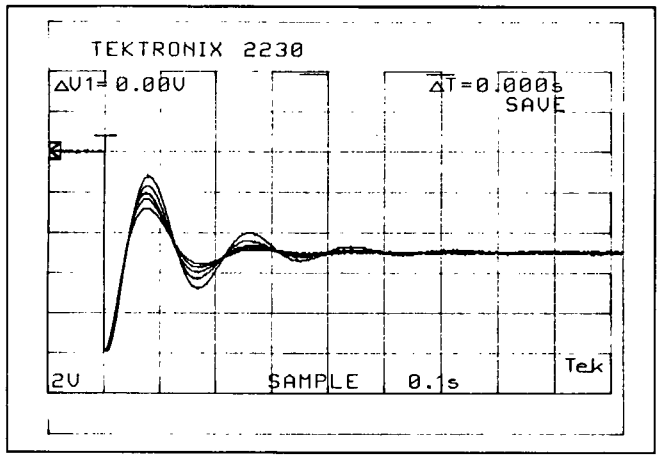
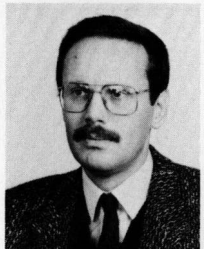
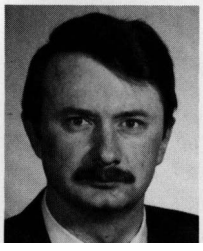


Figure 15. Multi rate multiprocessor real time simulation of closed loop servo system using SSI discretization technique

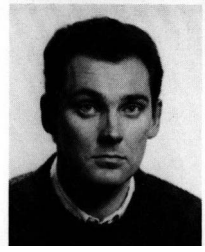
4. Cosic K., Miler I., Raseta I.: "Design and testing of homogeneous single bus tightly coupled multiprocessor system for real time simulation", *Informatica*, Vol. 13, No.3, 1989.
5. Cosic K., Kopriva I.: "Optimization of discrete time model for digital real time simulation", *Proceedings of the European Simulation Multiconference*, Part B, Rome, 1989.
6. Cosic K., Miler I., Kopriva I.: "Automatic discretization and code generation for real time simulation of linear time invariant modules", *Proceedings of the European Simulation Multiconference*, Nuremburg 1990.
7. Franklin, M. "Parallel solution of ordinary differential equations", *IEEE Transactions on Computers*, Vol.C–27, No.5, May 1987.
8. Hanselman, H. "Implementation of Digital Controllers – A survey", *Automatica*, Vol. 23, No. 1, 1987.
9. Hanselman, H., Schwartz, A. "Generation of fast target processor code from high level controller descriptions", *Proc. 10th IFAC World Congress*, 1987.
10. Howe, R.M. "Dynamics of real-time digital simulation", Lecture notes, Applied Dynamics International, Ann Arbor, 1988.
11. Howe, R.M. "Special considerations in real time digital simulation", *Proceedings of the Summer Computer Simulation Conference*, 1983.
12. Karplus, W.J., Makoui, A. "ALI: A CSSL/multiprocessor software interface", *SIMULATION*, August 1987, pp. 63–71.
13. Katz, P. "*Digital Control Using Microprocessors*", Prentice Hall, 1981.
14. Kempf, D.J., Bonderson, L.S., Slafer, L.I. "Real time simulation for application to ABS development", *Automotive International Congress and Exposition*, Detroit 1987.
15. Kopriva I. "The optimization of discrete time model for digital real time simulation", MS Thesis, Faculty of Electrical Engineering, Zagreb, 1990. (Original in Croatian.)
16. O'Grady, E., Pearse, Wang Chang-Hsein. "Parallel processor performance in a jet engine simulation", *Proceedings of the Summer Computer Simulation Conference*, 1984.
17. Palas, R.F. "Dynamic Simulation in System Analysis and Design", *Chemical Engineering Progress*, February 1989.
18. Palusinski, O.A. "Simulation of dynamic systems using partitioning and multirate integration techniques", *Proceedings of the Summer Computer Simulation Conference*, 1983.
19. Pimentel, J.R. "Real time simulation using multiple microcomputers", *SIMULATION*, March 1983.
20. Russell, K. "Power plant simulation – effective training for real–life emergencies", *Electrical Review*, April 1983.
21. Smith, J.M. *Mathematical Modeling and Digital Simulation for Engineers and Scientists*, John Wiley & Sons, 1977.
22. Zhang, S.X., Schroer, B.J., Tseng, F.T. "An automatic programming approach to simulating prelaunch activities", *SIMULATION*, July 1989.



KRESIMIR COSIC received his BS, MS, and PhD degrees in 1973, 1978 and 1984 respectively from the Faculty for Electrical Engineering in Zagreb. Currently he is associate professor at the Faculty of Electrical Engineering in Zagreb, where he lectures on digital control system design and digital real time simulation. He is author or co-author of 30 technical papers related to digital servo system design and real time simulation applied for "hardware–in–the–loop" testing and "man–in–the–loop" training. Dr. Cosic was interested in establishing the "Laboratory for Real Time Simulation" in Zagreb for educational purposes, industrial applications and research.



IVAN MILER received BS degree in electrical engineering from the High Technical School in Zagreb and the MS degree in digital real time simulation of dynamic systems using multiprocessor resources from the Faculty of Electrical Engineering in Zagreb. Currently he is senior lecturer at the High Technical School. His research interests are in the area of hardware and software for multiprocessor based digital control systems and real time simulation applied to "hardware–in–the–loop" testing and "man–in–the–loop" training.



IVICA KOPRIVA received his B.S. degree in Electrical Engineering from the High Technical School in Zagreb in 1987, and a M.S. degree from the Faculty of Electrical Engineering in Zagreb in 1990, both in the field of digital real time simulation of dynamic systems. During this period he was concerned with methods for discretization of linear systems and with multiprocessor based digital real time simulation. Since 1987 he has worked at RIZ Zagreb, Professional Electronic Department, on microprocessor controlled whip antennas watching units. His current interests include hardware and software environment for multi-microprocessor based workstation for simulator design.