

# ADAPTING POWER-SERIES INTEGRATION TO REAL-TIME SIMULATION

Robert Howe<sup>1</sup>, Michal Kraus<sup>2</sup>, Jiří Kunovský<sup>2</sup>

<sup>1</sup>University of Michigan, Department of Aerospace Engineering

<sup>2</sup>Brno University of Technology, Faculty of Information Technology  
Božetěchova 2, CZ-612 66 Brno, Czech Republic

*kunovsky@fit.vutbr.cz (Jiří Kunovský)*

## Abstract

Simulation of physical systems using digital computers continues to play an ever increasing role in all aspects of today's technological society. In general the basis for simulation resides in mathematical models of the systems being simulated. In the case of continuous dynamic systems these models consist of either nonlinear ordinary or partial differential equations. The simulation of these systems and hence the simulation of the corresponding mathematical models can be accomplished by numerical integration of the differential equations.

An original mathematical method which uses the Taylor series method for solving differential equations in a non-traditional way has been developed. Even though this method is not much preferred in the literature, experimental calculations have shown and theoretical analyses have verified that the accuracy and stability of the Taylor series method exceeds the currently used algorithms for numerically solving differential equations.

It is the aim of the paper to adapt power-series integration (Taylor series) to real-time simulation. In real-time digital simulation the numerical integration step size  $h$  is almost always fixed. The same is expected from corresponding power-series integration.

Actually, it may be difficult to apply the power series method to real time simulation, since the required higher derivatives of the real-time inputs will not in general be available. Furthermore, many real-time simulations involve derivative functions that are represented by multi-dimensional data tables rather than analytic functions. In this case the required state-variable derivatives do not exist.

Adapting power-series integration to real-time simulation in this paper is based on a model representation. The real system is driven by the model and all the control is specified in the model. Next step how to speed up simulation is to use a special digital hardware.

**Keywords:** power-series, real-time simulation, differential equation, controller

## Presenting Author's Biography

Jiří Kunovský graduated at Brno University of Technology, in 1967. During most of his time at BUT he has taught and directed research in Computer Science, specially in simulations of "Security-Oriented Research in Information Technology". He has created the simulation language TKSL (II-2007/TKSL is available now).



## 1 Modern Taylor Series Method

A large number of integration formulas have been published especially for solving special systems of differential equations. In general, it is not possible to choose the best method but for a subclass of tasks defined by similar properties the most suitable method could always be found. The "Modern Taylor series method" has proved to be both very accurate and fast. It is based on a direct use of the Taylor series.

The main idea behind the Modern Taylor Series Method is an automatic integration method order setting, i.e. using as many Taylor series terms for computing as needed to achieve the required accuracy. Methods of different orders can be used in a computation.

Obviously, the more Taylor series terms are used the higher is the achieved accuracy. The main problem connected with using Taylor series is the need to generate higher derivatives. This is in fact the reason why Runge-Kutta formulas of various orders have been used.

If we succeed, however, to obtain the terms with higher derivatives, the accuracy of calculations by Taylor series method is extreme (it is in fact only limited by the type of the arithmetic unit used). This is typical, in particular, for the solution of the technical initial problems. Technical initial problems are defined as initial problems where the righthand side functions of the system are those occurring in the technical practice, that is functions generated by adding, multiplying and superposing elementary functions. Such systems can be expanded into systems with polynomials on the righthand sides of the equations. In such a case the Taylor series terms can easily be calculated.

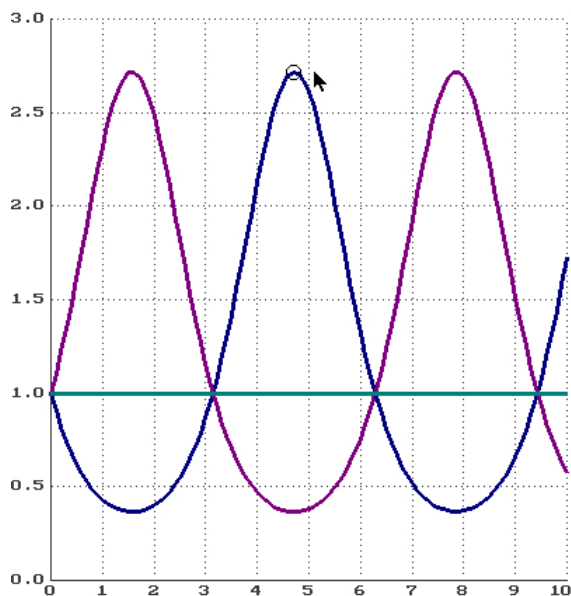


Fig. 1 Solutions for  $a = 1$

The Modern Taylor Series Method has been implemented in II/2007-TKSL software. The high accuracy

of the TKSL is demonstrated on the following system of equations

$$y' = a \cdot y \cdot \cos(t) \quad y(0) = 1 \quad (1)$$

$$x' = -a \cdot x \cdot \cos(t) \quad x(0) = 1 \quad (2)$$

$$z = x \cdot y \quad (3)$$

In Fig. 1,  $x$  (blue),  $y$  (violet) and  $z$  (green) as functions of time are shown in the course of the computation (for  $a = 1$ ).

The system of equations (1), (2), (3) was deliberately designed for the variable  $z$  to characterize the accuracy of the computation.

For the test function  $z = x \cdot y$  we have  $z = 1$  since the exact solution of (1) is  $y = e^{a \cdot \sin(t)}$  and the exact solution of (2) is  $x = e^{-a \cdot \sin(t)}$ .

The accuracy of the computation is preserved even if the variables reach values of  $10^{443}$  and  $10^{-435}$  by order of magnitude. The numerical solution of the system (1) (2) reaches these values for  $a = 1000$  (Fig. 2).

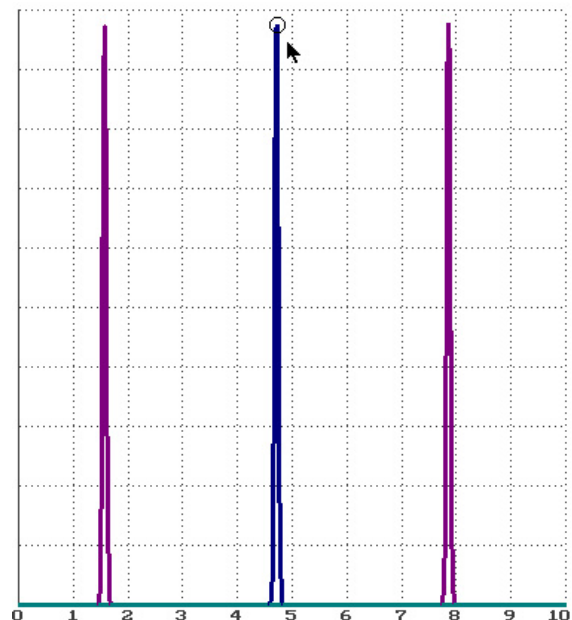


Fig. 2 Solutions for  $a = 1000$

However, since the Modern Taylor Series Method (power series method) is a single-pass method, it is possible that it will execute much faster per overall integration step than the multiple-pass Runge-Kutta methods.

On the other hand, if the derivate function  $f = dx/dt$  is a complex analytic function, the calculation of the derivatives of  $f$  may be very intensive, especially for a large order  $k$  of integration algorithm. This will slow down the power series method, so that it is not easy to draw general conclusions on which method will be faster.

The speed tradeoff will be very problem dependent. Note also that it may be difficult, if not impossible, to

apply the power series method to real time simulation, since the required higher derivatives of the real-time inputs will not in general be available. Furthermore, many real-time simulations involve derivative functions that are represented by multi-dimensional data tables rather than analytic functions. In this case required state-variable derivatives do not exist. Nevertheless, power series integration methods can be very efficient for simulating particular types of differential equations.

## 2 "Real-time" system P

There are many examples of digital simulations involving real-time inputs and outputs, including spacecraft simulators, land vehicles simulators, ships simulators, process control simulators, power plant simulators, etc. As in the flight simulator, the hardware in the loop may be a human operator, in which case the simulator can be used for mansystems development and evaluation, or for the training of human operators.

As an example, first order differential equations (4), (5)

$$u' = 500 \cdot z - 51 \cdot u - 50 \cdot y \quad u(0) = 0 \quad (4)$$

$$y' = u \quad y(0) = 0 \quad (5)$$

$$z = 1 \quad (6)$$

describe the "school" system P. Function  $y$  (green) together with time function  $u$  (violet) and order of method  $ORD$  (blue) is in Fig. 3.

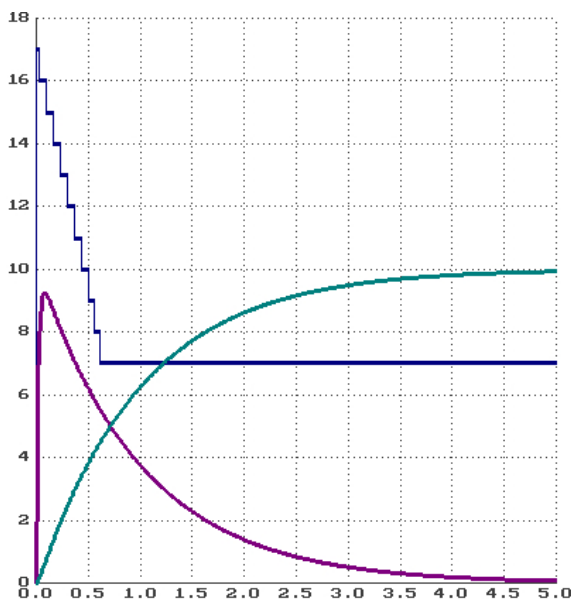


Fig. 3 Time functions  $y, u, ORD$

## 3 Closed-loop controller 1

A common closed-loop controller architecture is the PID controller. The output of the system  $y(t)$  is fed back to the reference value  $r(t)$ . The controller  $C$  then takes the error  $e$  (difference) between the reference and the output to change the inputs  $u$  to the system under

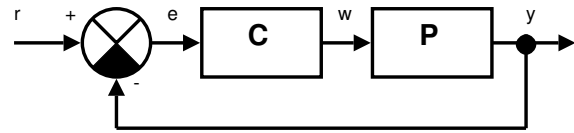


Fig. 4 Closed-loop controller 1

control  $P$ . This is shown in the Fig. 4. The Fig. 4 is described by

$$e = r - y$$

$$w = k_p \cdot e + k_i \cdot \int e + k_d \cdot \dot{e}$$

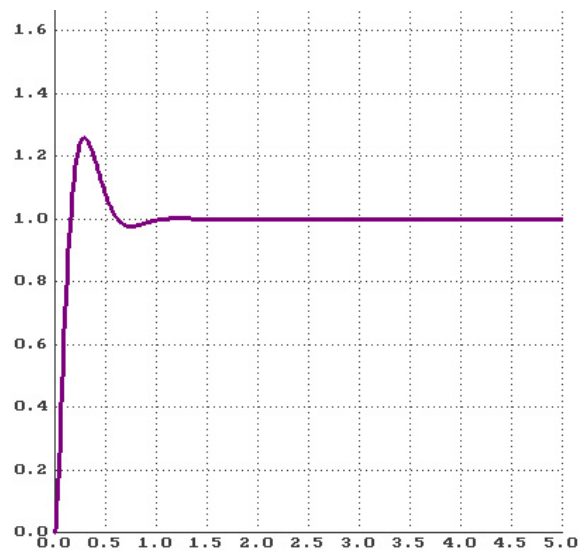


Fig. 5 Solution  $y$

An example of the solution  $y$  for  $k_p = 0.9, k_i = 0, k_d = 0, r = 1$  is in Fig. 5.

## 4 Closed-loop controller 2

Similar example for the closed-loop controller architecture in Fig. 6 is described by

$$e = r - w$$

$$w = k_p \cdot y + k_i \cdot \int y + k_d \cdot \dot{y}$$

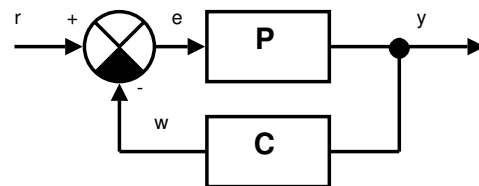


Fig. 6 Closed-loop controller 2

Solution of  $y$  for  $k_p = 0.9, k_i = 0, k_d = -0.05, r = 1$  is in Fig. 7.

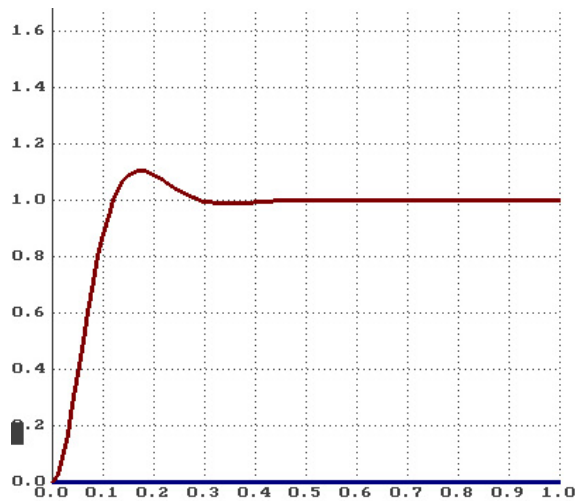


Fig. 7 Solution  $y$

### 5 Model representation

Adapting power-series integration to real-time simulation is based on a model representation (Fig. 8).

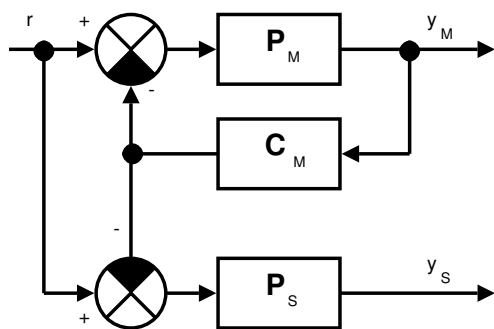


Fig. 8 Model representation

$P_M$  is the "Taylor" model of the real system  $P_S$ . Controller  $C_M$  generates proportional, integration and derivate part of  $y_M$ .

The real system  $P_S$  (using either 1st, 2nd, 3rd or 4th order integration formulas) is driven by the model and all the control is specified in the model. Model  $P_M$  uses Taylor series integration method (power series integration).

Actually, this is the main idea of adapting power-series integration to real-time simulation - complicated required high order derivatives are not computed from the real system  $P_S$  but they are indirectly generated from the model ( $P_M, C_M$ ).

If parameters of the model and the real system are the same, step responses  $y_M, y_S$  are the same, of course, and corresponding curves are the same as in Fig. 7.

Fig. 9 and Fig. 10 show the step response and error  $Err$  for different perturbations of a real system ( $Err = y_M - y_S$ ).

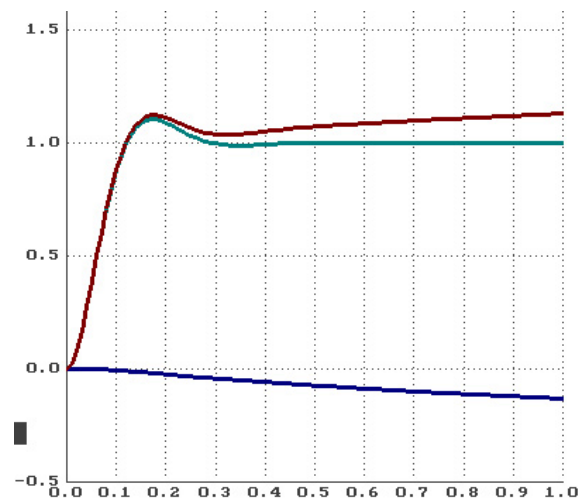


Fig. 9 Different parametrs of the model and real system

Blue function in Fig. 9 represents error of solution  $Err$ , brown function gives step response (coefficient 50 in equation (4) has been changed to value 40), green required function gives step response described in Fig. 7.

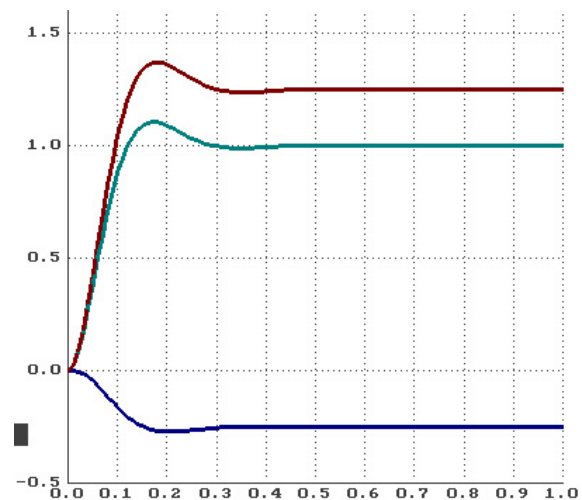


Fig. 10 Different parametrs of the model and real system

Similarly, blue function in Fig. 10 represents error of solution  $Err$ , brown function gives step response (coefficient 50 in equation (4) has been changed to value 40 and coefficient 51 in equation (4) has been changed to value 41) and green required function gives step response described in Fig. 7.

Final representation of adapting power-series integration to real-time simulation is in Fig. 11. A special feedback  $C_{KKI}$  in real system derived from difference  $Err$  gives us a possibility to eliminate different coefficients of model and real system (Fig. 12). Again, blue

curve represents  $Err$ ; different parameters are those described in Fig. 10.

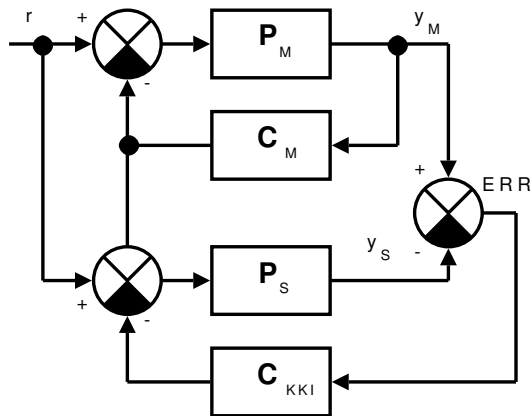


Fig. 11 Final model representation

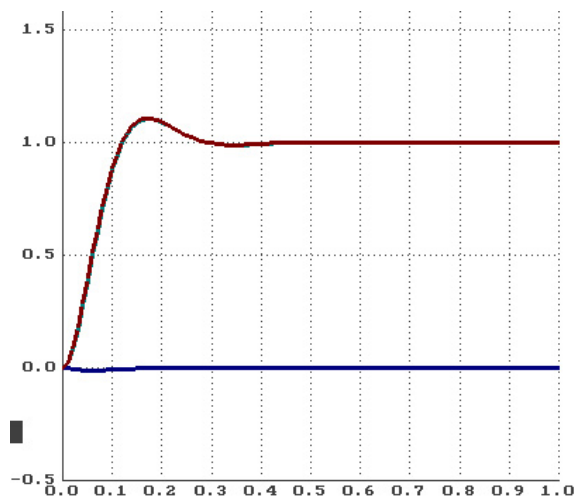


Fig. 12 Result of feedback  $C_{KKI}$

## 6 Digital-to-analog and Analog-to-digital conversion

Some of the inputs and outputs for the model representation in Fig. 11 are in continuous or analog form. In this case the continuous inputs must be converted to digital form using A to D (Analog to Digital) converters. The output of the A to D converter is then a data sequence, usually with a fixed time interval  $h$  between samples of the analog input. In this case  $h$  usually (but not always) becomes the step size for the numerical integration. Similarly, the outputs, in the form of data sequences, are converted to continuous form using D to A (Digital to Analog) converters. Influence of D to A and A to D will be discussed during the conference, together with a possible corresponding time delay.

## 7 Taylor Series Numerical Integrator

It should be noted that integration method uses a fixed integration step size because of the real-time requirement. It may also be true that the use of very efficient fixed-step integration algorithms may actually speed up many simulations.

Next step how to speed up simulations is to use a special digital hardware, mainly specially designed arithmetic logical units.

We outline the design and implementation of an FPGA-based numerical integrator that will form the basis of our FPGA-based parallel hardware. Currently, our processor uses the Taylor series numerical integration algorithm to solve the ordinary differential equations and can be used as a part of  $P_M$  and  $C_M$  for final model representation (in Fig. 11).

The basic part of our system is the arithmetic logic unit (ALU) designed for Taylor series numerical integration algorithm. We have analyzed mathematical operations and following operations are required:

- addition
- subtraction
- multiplication

The architecture of our fixed-point processor is "serial" in communication and "parallel" in computation (Fig. 13).

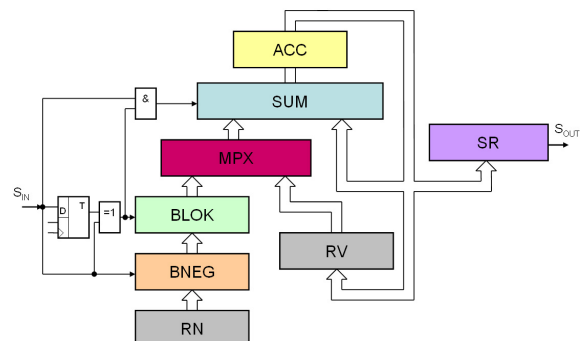


Fig. 13 Arithmetical logic unit

Our numerical integrator consists of the following well-known blocks:

- SUM - Summer
- ACC - Accumulator
- MPX - Data multiplexer
- RN - Registr of integration step
- RV - Registr of initial condition
- BNEG, BLOK - Circuits for Booth algorithm (multiplication)

- SR - Communication registr
- Additional necessary logic

A field programmable gate array (FPGA) is used for the design of our processor. Some results of testing of our processor are in Fig. 14, for those interested in the detail function.

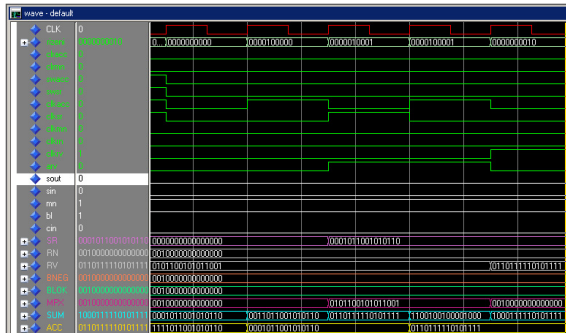


Fig. 14 Simulation of ALU

## 8 Summary

All computations have been completed by II/2007-TKSL software.

Details together with influence of AD, DA converters and more sophisticated system  $P_M$  will be presented during the conference.

We will continue to study the application of power-series integration to real-time simulation and continue to discover major problems and advantages with it. It appears to us that the main attraction of power-series integration for non-real-time simulation is the large integration time step.

**Acknowledgment.** *The research has been supported by the project MSM0021630528 (Ministry of Education, Youth and Sports, Czech Republic) "Security-Oriented Research in Information Technology".*

## 9 References

- [1] R. Howe. Dynamics of Real-Time Simulation. *Applied Dynamics International*, Michigan, 1998.
- [2] J. Kunovsky. Habilitation work. *Brno University of Technology*, 1995.
- [3] B. L. Stevens and F. L. Lewis. Aircraft control and simulation 2nd edition. *Wiley*, ISBN 0-471-37145-9, 2003.