# Advanced TKSL

Pavel Pospíšil *
pospispa@dcse.fee.vutbr.cz

Petr Sezemský*
sezemsky@dcse.fee.vutbr.cz

Miloň Papežík*
papezik@dcse.fee.vutbr.cz

## Abstract

This paper aims at TKSL research and future advanced version of TKSL. It summarizes advantages and disadvantages of TKSL/Pentium, the latest TKSL version and gives summary of resent TKSL research. Finally, it sets next TKSL version requirements.

## 1 Modern Taylor Series Method

By a numerical solution of an ordinary differential equation

$$y' = f(t, y), \qquad y(t_0) = y_0 \tag{1}$$

we understand the finding of a sequence:

$$[y(t_0) = y_0], \quad [y(t_1) = y_1], \quad [y(t_2) = y_2], ..., [y(t_n) = y_n].$$

The best-known and most accurate method of calculating a new value of a numerical solution of a differential equation (1) is to construct the Taylor series in the form

$$y_{n+1} = y_n + h * f(t_n, y_n) + \tfrac{h^2}{2!} * f^{[1]}(t_n, y_n) + \ldots + \tfrac{h^p}{p!} * f^{[p-1]}(t_n, y_n), \tag{2}$$

where $h$ is the integration step.

Methods of numerical solutions of differential equations have been studied since the end of the last century. A large number of integration formulas have been published especially for solving special systems of differential equations. In general, it was 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 presented " Modern Taylor Series Method " has proved to be both very accurate and fast. It is based on a direct use of the Taylor series. Detail explanation of the Modern Taylor Series Method including many practical examples can be seen in [1].

* Department of Computer Science and Engineering, Faculty of Electrical Engineering and Computer Science, Technical University of Brno, Božetěchova 2, CZ-612 66 Brno, Telephone: +420-5-7275283 (+420-5-41212219), Fax:+420-5-41211141

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.

## 2 The Simulation Language TKSL/Pentium

Theoretical work on the numerical solution of ordinary differential equations by the Taylor series method has been going on for a number of years. The simulation language TKSL/Pentium has been created to test the properties of the technical initial problems and to test an algorithm for Taylor series method.

TKSL/Pentium has the following features

- user friendly environment (Turbo Vision),
- adjustable computation accuracy,
- adjustable method order,
- computation with variable integration step h,
- exact detection of discontinuities.

TKSL offers the most practical features available today and it also presents an entirely new approach to solving continuous systems:

- on analysing a particular continuous system only the required precision of the solution is defined,
- the implicitly built-in integration method of order 64 (with the integration step h automatically defined/fixed) is distinctly superior in the computation quality to other integration algorithms currently used,
- the order of the implicitly built-in integration method can interactively be increased (yielding a new computation quality),
- stiff systems are viewed as systems of ordinary differential equations (and as such are automatically solved with the required precision).

### 2.1 Programming in TKSL/Pentium

Programming in TKSL/Pentium is very easy. The menu system in TKSL/Pentium is very similar to that in TURBO PASCAL 6.0. Using TKSL/Pentium is demonstrated on the equation:

$$y' = aycost, \quad y(0) = 1, \quad a = 2.$$

The corresponding source text in TKSL/Pentium is

```
var y;
const a=2,
    tmax=10,
    dt=0.2,
    eps=1e-20;
system
    y'= a*y*cos(t)    & 1;
sysend.
```

All variables that will be needed are declared in the line starting with var. All necessary constants are declared in the line starting with const ( a is a constant in the equation, tmax  is the maximal computation time , dt  is the step size and eps is the required accuracy).

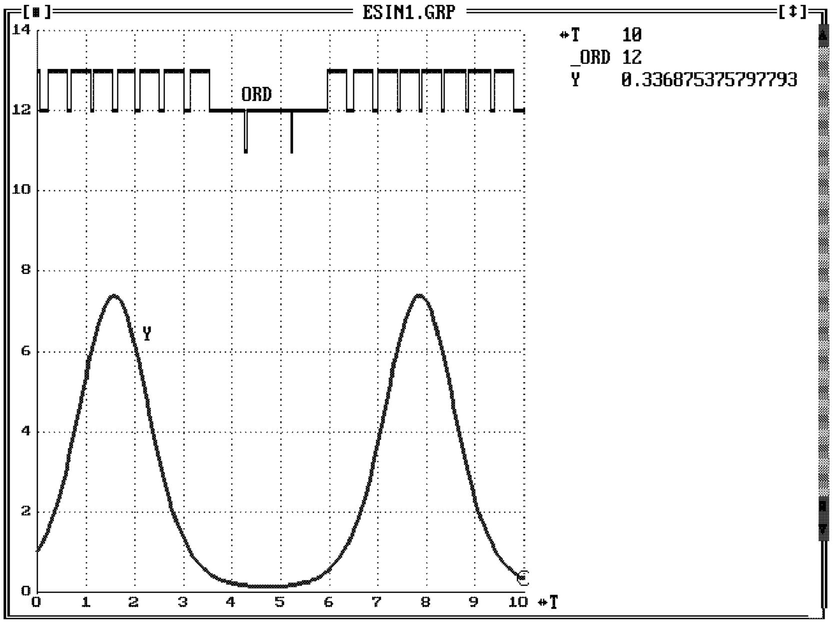


Figure 1:

The equation (3) is declared in block mode (between system and sysend). The initial condition for differential equation (3) is written in the form & initial condition.

The two typical windows which are displayed on the screen immediately after the program is started can be seen in Fig.2.1. In the right-hand part of Fig.2.1 the symbol $\leftrightarrow$ marks the variable which is plotted on the horizontal axis (in this case the time T).

The variable T is increased by the preset step size dt and at the same time the corresponding values of the variables ORD and y (ORD stands for the method order) are shown in the right-hand part of Fig.2.1. In the left-hand part of Fig.2.1 two functions of time are shown in the course of the computation. There are graphs of y and ORD . The last function has not yet been published in any paper on this subject.

In the above example ( for a=2, integration step h=0.05s ) the value of ORD ranges between the values 11 and 13.

If the value of integration step increases to h=0.1s , the corresponding resulting solution of the equation (3) (again for a=2) is in Fig.2.2 . The value Y of the numerical solution at time T=10s is of course the same, only the value of ORD, as expected, has increased ( ORD ranges between 13 and 16).

## 3 The Multibyte Arithmetic

The numbers over which the arithmetic operations are performed are in a floating-point representation:

$$x = M.z^E$$

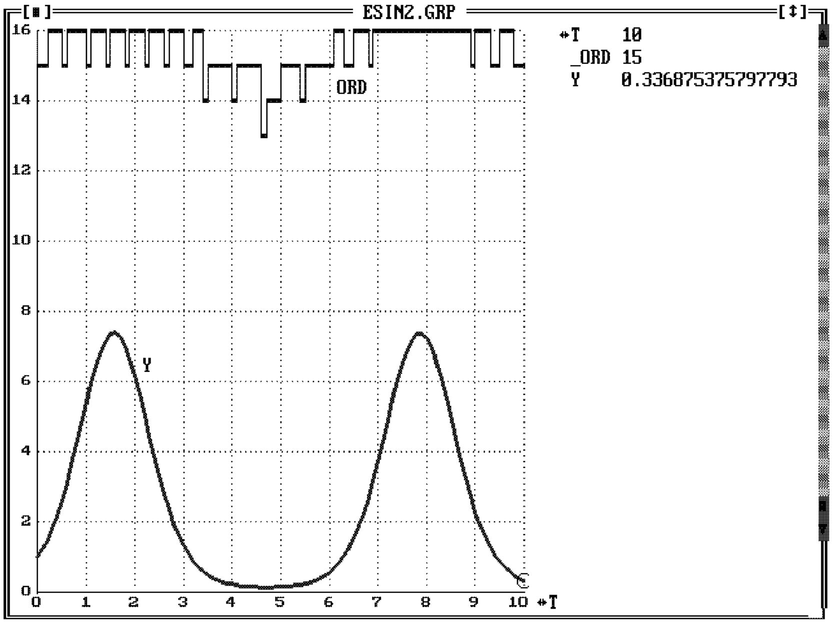


Figure 2:

Their accuracy is determined by the number of bits used for encoding the mantissa. A module has been developed enabling to select an arbitrarily great number of bits for encoding the mantissa. It is virtually limited only by the available memory size. It is easy to see how accurate the calculations with such a tool are especially if we consider that with a mathematical co-processor the maximum length of the mantissa is only 64 bits. On the other hand it must also be taken into account that the increased accuracy is traded off by the time needed to perform the mathematical operations.

To use the multiple arithmetic for calculating the solution of homogeneous differential equations it is sufficient to implement the following operations: addition, subtraction, multiplication, division.

Many experiments have been done to find and prove advantages of the multibyte arithmetic. It was shown (see [2, 4, 5]) that the multibyte arithmetic is very useful for reducing rounding error. Usage of the multibyte arithmetic is necessary for a group of scientific problems.

## 4 Polynomial Transformations

The main problem connected with using Taylor series (in the form of (2)) is the need to generate higher derivatives$f^{[1]}, f^{[2]}, \ldots$ . 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, of the solution of the technical initial problems.

Technical initial problems are defined as initial problems where the right-hand 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 right-hand sides of the equations. In such a case the Taylor series terms can easily be calculated.

To demonstrate this, we analyse the equation:

$y' = aycost, \qquad\qquad y(0) = y_0$

A simple computation scheme based on equation (1) follows:

$$f(t,y) = aycost.$$

Let $\quad v = cost.$

Then

$$\begin{aligned}
f(t,y) &= ayv, \\
f^{[1]}(t,y) &= a(f(t,y)v + yv'), \\
f^{[2]}(t,y) &= a(f^{[1]}(t,y)v + 2f(t,y)v' + yv''),
\end{aligned}$$

.

.

$$f^{[p-1]}(t,y) = a\left(\sum_{i=0}^{p-2} f^{[p-2-i]}(t,y)\ v^{[i]}\ \binom{p-1}{i} + yv^{[p-1]}\right), \qquad (\quad p \geq 2),$$

where

$$\begin{aligned}
v' &= -sint, \\
u &= sint, \\
v^{[p]} &= -v^{[p-2]} \qquad (\quad p \geq 2), \\
u^{[p-1]} &= v^{[p-2]} \qquad (\quad p \geq 2).
\end{aligned}$$

Similar constructions can be created for all elementary functions, such as exp, sin, cos, tg, cotg, ln, sinh, ....

The idea above requires software capable of automatically performing the decomposition of the right-hand sides of ordinary differential equations.

This approach has been implemented in a simulation language TKSL/Pentium ( an implementation of the Taylor Kunovsky Simulation Language on an Intel Pentium based personal computer). The system of differential equations is transformed into a form of algebraic-differential equations that allows straight usage of the Taylor series.

A new approach of polynomial transformations has been developed (see [3]). In this case, the system of differential equations is transformed into a form of differential equations with polynomial right-hand side. This approach is suitable for straight Taylor series usage but it is also possible to compute derivatives numerically and quickly that can be used for a new step size control.

# 5 An Advanced Step Size Control

The TKSL/Pentium computes terms of Taylor series expansion straightly. So if a chosen step size is too big we have not only to decrease the step size but we also have to throw away all computed terms of Taylor series expansion. In this case we have lost too much effort.

The advanced step size control proposes fast numerical computation of derivatives that is allowed by polynomial form of equations. After that we compute terms of Taylor series expansion and the solution. The most time consuming part of computation is the computation of derivatives that has quadratic complexity while other parts has just linear complexity. It is obvious that if we choose too big step size we do not have to recalculate derivatives but we only have to choose appropriate step size and recalculate the solution that is much easier and faster than a complete recalculation.

Computation of derivatives do not only saves the spent effort but it allows derivatives investigations. Once the derivatives are computed the user can study behaviour of derivatives that can help him in better understanding to the solved problem.

## 6 TKSL/Java

Next step in TKSL development is to create a system with user-friendly graphic editor suitable to drawing analogue schemes. The analogue scheme which represents a system of differential equations is internally converted to the simulation language. Now the simulation can be run and the results can be shown in graphs as in TKSL/Pentium system. But not only simulation. The system is capable to generate, compile (using existing compilers) and run programs in C language. These programs should use the multibyte arithmetic to improve accuracy. A great deal of numerical methods including Taylor series method is built in the system. The program can be adapted to any machine in order to exploit its special properties (for example more processors).

The graphic interface of TKSL/Java is made in Java language which ensures system independency and an user-friendly interface.

## 7 Conclusion

The features of TKSL/Pentium were presented. Authors suggest following improvements:

- Java usage so that the TKSL/Java will be system independent and will have a user-friendly interface
- Advanced step size control usage that will bring a faster computation
- Multibyte arithmetic usage that will allow more accurate computation
- Preservation of all TKSL/Pentium advantages

Authors will kindly welcome all suggestions and advices of other possible modern features and improvements of TKSL/Java.

## References


1. Kunovský, J.: Modern Taylor Series Method, Habilitation work, FEI VUT Brno, 1995
2. Kunovský, J., Zbořil, F.: Optimum Differential Expression in Solving PDE, In: Proceedings of 31st Conference "Modelling and System Simulation MOSIS'97", pp. 80–85, ISBN 80-85988-16-X, CZ, Hradec nad Moravicí, 28.–30. April 1997
3. Mikulášek, K.: Transformations of systems of differential equations to polynomial form and their applications, Themes for Dissertation Thesis, FEI VUT Brno, 1998
4. Papežík, M., Pospíšil, P., Němec, V.: Simulation System TKSL/MA, In: Proceedings of 31st Conference "Modelling and System Simulation MOSIS'97", pp. 221–226, ISBN 80-85988-16-X, CZ, Hradec nad Moravicí, 28.–30. April 1997
5. Pospíšil, P.: Taylor Methods for ODE's, Diploma Thesis, DTU Lyngby, 1996