DE GRUYTER OPEN

Print ISSN: 1312-2622; Online ISSN: 2367-5357
DOI: 10.1515/itc-2015-0010

Solving Nonlinear Ordinary Differential Equations in .NET Framework Environment with X-MPIR

V. Dzhambov

Key Words: *High precision computation; nonlinear ordinary differential equations; computational mathematics; NET Framework.*

Abstract. *This paper concerns high precision numerical computing of nonlinear ordinary differential equations in .NET Framework environment. It describes a part of the research undertaken for the purpose of creating a tool library for arbitrary precision calculations and may be concerned as a continuation of the beginning in this direction, previously described in [10], including special functions computing. Some possible applications are described on the basis of the extensions, added to the library. A brief description is given of the library part, connected with the extensions being added.*

1. Introduction

Arbitrary precision computations are not a self-purpose. They are related to receiving precise values when solving mathematical models in different areas, including for instance, non-linear dynamic systems. In [10] an announcement is made for creating a high precision computation library in the environment of .Net Framework. The application SFCLC has been represented there, illustrating a part of the possibilities of the library being created. Some extensions of this library are further described, discussing the solution of systems of ordinary nonlinear differential equations and the capacities they provide.

One of the stimuli for such an extension is that a possibility could be provided within the frame of the library being created for investigation of interesting nonlinear models. More particularly, for a dynamic system the quantitative investigation with very high precision of the solution is a must [11,12], connected with the chaotic behavior for a finite time interval. Taking in mind this requirement, an extrapolation method of variable order of the accuracy (Gragg-Burlish-Stoer algorithm) may be used; for any details refer to [3]. A desirable additional requirement is the preservation, if possible, of the model structure by the numerical methods for its solution. Under these conditions the selection of Gauss-Legendre implicit Runge-Kutta methods is appropriate. They are symmetric, sypmplectic and A-stable for all orders. For a description of Runge-Kutta implicit schemes and their properties refer to [1,2] and [5].

One of the purposes is to demonstrate that very useful and mutually complementing computational tools, solving non-trivial problems with high precision computations may be implemented in a concrete environment, which is wide spread for personal computers, but this fact is underestimated by the software developers of scientific applications. There are certain reasons for this, of course, but in our opinion, there are advantages also: comparatively easy integration in various functionalities, such as visualization and interactivity in one and the same application if, of course, the necessary methods for solving the concrete problems are available, which is the purpose of the library being created. Of course we keep in mind that the creation of a complete system, covering a great range of tools for solving various mathematical models, is a too ambitious task at least due to the huge amount of the work required. But the implementation of some basic tools enables the interested investigator to create own applications in the given environment, exactly matching his concrete interests; to design, instead of be a user of a ready system. Another purpose we aim at is to provide such a possibility.

2. General Structure and Extensions of the Library Concerning the Solution of ODEs Systems

The arbitrary precision of the calculations (with real numbers) is provided by the library X-MPIR. X-MPIR ensures the binding from C# to MPIR (for details search the Web, there is no independent site, devoted to X-MPIR at this moment). The library is grouped in several files that we shall briefly describe. Only the components, referring to the problem solved, will be mentioned.

funcs.cs: Basic implementation of the functions and constants, described in [10], combined in a namespace MPMath_R, most in the public static class MPMath, and several general numeric methods, plus a class solving nonlinear ODE systems of public class IRK_Solver with the help of an implicit Runge-Kutta scheme. There is a variant of this solver, working with double, namely public class IRK_Solver_d. It proves to be useful in cases when only a qualitative picture is necessary, so that the scheme can be used at not very high order, that speeds up the calculations. Besides these variants of solving ODE systems, based on Runge-Kutta implicit scheme, an extrapolation method is implemented (class ODEX). The well known ODEX Fortran program [3,4] is taken as a basis. It is adapted for C# + XMPIR, its parameters being set for arbitrary accuracy. More specifically, 'km' and 'uround', as well as the current calculation precision are determined depending on

the requirements of the needed accuracy of the solution 'tol'. Besides this, the behavior of the parameter iout=1 is altered, so that output points are obtained within an interval, not bigger than a desired value (new input parameter). This is implied since this value of 'iout' is basic, when the required accuracy is high. In this case the use of iout=2 (dense output) leads to very small values of the step *h*, due to the built in estimation of the error at interpolation.

The remaining files include the extensions.

funcs_c.cs: It accomplishes complex arithmetic of arbitrary precision with the help of an internal class MPComplex, and the functions, existing in funcs.cs are implemented in the namespace MPMath_C, public static class CMPMath,. It contains also a part of the methods and the complex variant of the solver of ODE nonlinear systems IRK_Solver_C. An integral logarithm is added as well, an integral exponent and the elliptic Jacobi functions.

mpir_parse_fx.cs, mpir_parse_cx.cs: Variants of a parser for mathematic expressions, supporting arithmetic operations and the functions and constants, implemented in the previous two files. If no errors are available, the parser creates the converted in a postfix form list (List<Token>) from the elements of the expression, each one possessing specific information about the list element. In Token class there are fields foreseen for this information: a mathematic or literal constant, variable, arithmetic or functional operation, text presentation, priority, number of arguments, type of associativity, eventual value. This list is used by a concrete application, invoking the parser, in order to form a list of simplified elements (struct SToken {public int opcode; internal mpir.mpf_t val} or probably internal MPComplex val), including the operation code only, which may be updated by the given application in the process of the problem solution. This list is already prepared for interpretation with the help of a well known algorithm. The parser is designed to accept more than one variable and it denotes as parameters all variables that are not names of mathematic constants or name of the basic variable of the model being solved. This gives additional flexibility in the model setup that will be explained further on. If an error occurs, the parser acknowledges the type and position of the error in the expression.

Even using only a part of the tools in the first two files above described, one can create console applications for certain problems or useful auxiliary applications with graphic interface, such as SFCALC, presented in [10]. The two latter enable the design of applications with considerably more user friendly interface, allowing the solution of a given class of problems without changing the program code, additional compilation, etc. Some of the possible applications of both types are described below.

SFCALC application may be considered a part of the library. Besides some new features, it has got a very simple, but important possibility to transfer the calculated result into the clipboard, as well as the inverse operation of taking a number, stored in a text form from the clipboard. This is especially appropriate for applications with graphical interactive interface, when the model given contains parameters.

3. The Meaning of Calculations with High Precision for Deterministic Models of Chaotic Behavior

We shall consider a deterministic system in an exactly determined meaning, namely a smooth dynamic system. The monographs and papers, referring to the chaotic behavior of dynamic systems, and to the dynamic systems as a whole, studied in different aspects, are quite numerous. The references cited [13-24] are just a small part, providing information about the questions being discussed in this section. Our purpose here is to focus on the smooth non-conservative dynamic systems with chaotic behavior.

Two questions are regarded herein. The first one: In case the model is not abstract, but connected with real phenomena in any area, like physics, chemistry, biology, economy, sociology, etc., to what extent must it be close to reality, so that calculations of high precision will be required, at least in a given interval? The second one: Are the mathematical tools, used in modeling and solution adequate enough? We leave aside the philosophic question what in fact we are investigating – any reality or our own consciousness, creating reality models, which is not apriori one and the same.

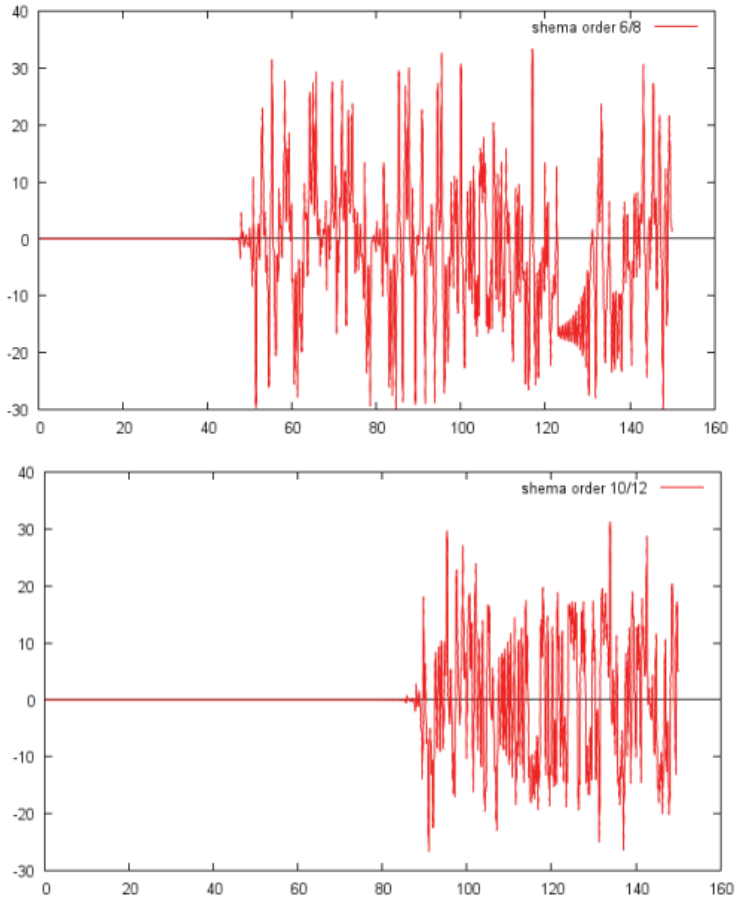
The two questions are connected to some extent. The first one could be answered, though rather conditionally. It depends on the position of its consideration. From a mathematical viewpoint, there are no problems, at least in the formulation. From a physical point of view, the smooth dynamic systems with or without chaotic behavior are not very exact. For example, the classical models of certain physical phenomena cease to be accurate for dimensions, co-measurable with those, typical for the matter discrete nature. This is from a formal viewpoint. On the other hand, it is not necessary to have an absolutely exact model in order to reflect correctly the main properties and qualitative features of the system behavior that happens quite often. The second question is more interesting. We must decline to the features of the quantitative solution of similar models. What characterizes them? Without any consensus on the exact definition of the chaos in systems of chaotic attractor, however there are two properties, which are considered essential: 1) Sensitivity to the initial conditions; and 2) Presence of a topologically transitive orbit in the attractor – i.e., an orbit passing through an arbitrary small neighbourhood of an arbitrary point of the attractor. Roughly put, the first condition is necessary, but not sufficient, while the second one approaches to giving a sufficient condition. Some authors add the requirement for availability of a dense set of periodic orbits. Researching many models from the positions of symbol dynamics, this is true, but it is nor clear whether this pattern is not a sequence of the first two. Refer to [24] to see the definitions, proposed by different authors.

In the context of real calculation of the model, the presence of a topologically transitive orbit there causes the impossibility for exact calculation of the orbit, taking in mind the finite accuracy of the calculations. No matter how high the calculations precision is, since it is finite, there comes inevitably a moment when the orbit calculated and this topologically transitive orbit are indiscernible for the computing process. The famous shadowing lemma is a reflection of this fact to some extent. Finally, are the calculation tools adequate? The answer again depends on the viewpoint. From a mathematical point of view there is no problem even if we consider constructive real numbers only. They are reproducible with arbitrary accuracy and either no question is set about the computing resurces, or it is idealized. From a practical viewpoint connected with real calculations though, the situation is different. The computing systems can potentially give arbitrary, but still finite accuracy of any constructive real number, but this is not sufficient for tracing of the exact orbit, due to the reason above described. We could mention here that in some popular expositions the notion „chaos of a deterministic system“ is often connected with the impossibility for exact prediction. This assertion needs elaboration. The question is in passing between the tools, formulating the model (ideal) and those available for approximate calculation (real). Yes, all calculations are with a finite number of symbols and this is limiting. If the question is thus put, we are not able to calculate the orbits even because of the fact that we cannot set exactly their initial point, for example one of the coordinates may be $\sqrt{2}$. Besides this, in some cases the prediction is somewhat relative, even if concerning the qualitative aspects of any process only. We cannot be certain whether a complex physical process is not periodic, because we cannot trace it to infinity. These difficulties suggest that at least a part of the important properties of a given dynamic system are more general, global, beyond the set of the individual orbits and this is true indeed. However, this is another topic – in this paper we discuss the calculation of these orbits in an exactly determined environment, which is also of interest in some cases.

To illustrate the practical situation, the calculation of the well known Lorenz system is given:

$$(1)\quad \begin{aligned} \dot{x} &= \sigma y - \sigma x \\ \dot{y} &= rx - y - xz \\ \dot{z} &= -bz + xy \end{aligned}$$

with standard selection of the coefficients $\sigma = 10$, $b = 8/3$, $r = 28$. As an initial point the following is chosen (10.6451, 4.06125, 36.057) – close to the attractor, so that the transition process is not computed. The following graphics show the difference of x in calculation with the help of the main solver in the library, implementing implicit Runge-Kutta scheme of different orders. The scheme order is indicated in the right corner of every graphic, respectively 6/8, 10/12, 14/16 and 18/20. All calculations are performed with a step of 0.01 and accuracy up to 120 decimal characters for 15 000 steps. For y and z the picture is similar.



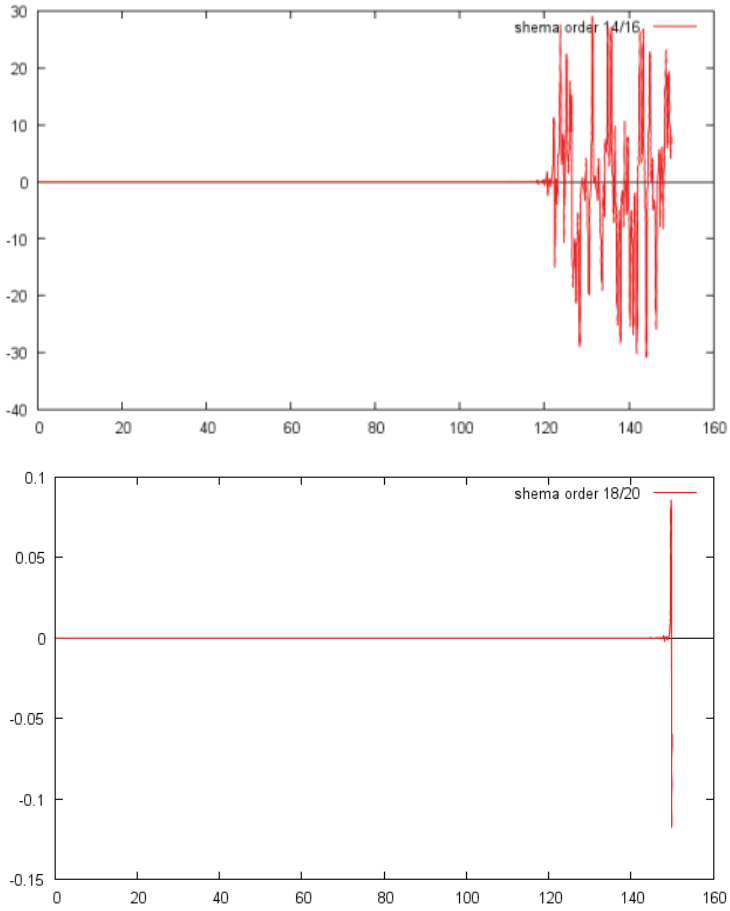


Figure 1. Accuracy of the solution for different scheme orders

4. Console Applications. Illustrations of Digital Calculation of Some Simple Smooth Dynamic Systems with Chaotic Behavior

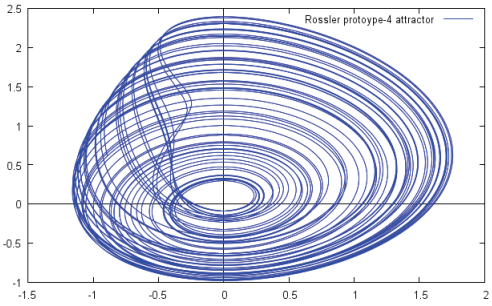
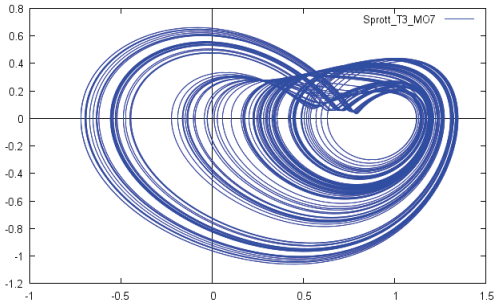
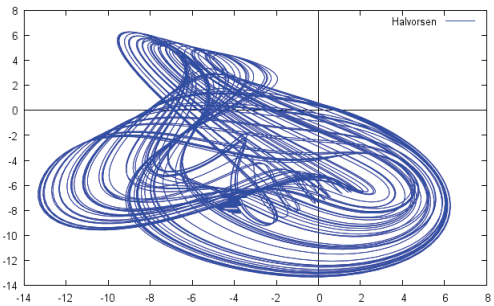
Three example systems with dimension 3 are given and the corresponding graphical presentations of the solutions, projected on the plane *xy* are indicated in *table 1*. The examples have been taken from Chapter 3 – Autonomous Dissipative Systems of [25]. In fact, with exception of Chapters 8 and 9, which require other calculation tools, the remaining examples in this book are accessible for visualization and have been used as a test. The solutions are computed by the main solver for 10 000 steps with a step length of 1/25 and order 4 of the implicit Runge-Kutta scheme.

5. An Application Example with Graphical Interface, VODE2

The program is designed for numerical study of ordinary differential equations of second order. It solves a problem with initial values $z = z_0$, $u_0 = u(z_0)$, $u'_0 = u'(z_0)$ (Initial Value Problem). The equations can be in the real or complex domain. The generated solution is shown graphically in the process of solving. The expression for the equation is given in a text form. A possibility to save the current state of the problem is provided, as well as reading the previously saved problem from a file. The program can also work in the so called expression mode, that is, computing and showing an expression (functional). In both modes additional parameters are allowed.

The program is implemented using C# in .NET Framework. It is built targeting the maximal possible portability (32- or 64-bit Windows systems). No installation is needed. The necessary files are copied into a separate folder. Any intermediate calculations are done with complex numbers. For all previously needed data, concerning the problem solving, the corresponding elements of the interface are provided. The formal correctness is verified. Depending on the stage of the problem formulating/solving, a different part of the interface is accessible. In the process of solving it is possible to reach a singular point. In this case a message is displayed „Apparently a pole reached“, which is not quite correct, because the point may be essential singularity, not a pole, but gives an idea that the problem has occurred. In this case the solving process terminates. The visualization is performed by redirecting the standard output to the previously started in a separate process program gnuplot. Commands are passed to this output for gnuplot to visualize a curve with the computed up to the moment

Table 1. Examples of simple dynamical systems with chaotic behavior

Rössler system, prototype 4 $\begin{cases} \dot{x} = -y - z \\ \dot{y} = x + ay \\ \dot{z} = b + z(x - c) \end{cases}$ $(a, b, c) = (0.5, 1, 3)$ $(x_0, y_0, z_0) = (2, 0, 1)$	
Memory oscillator $\dddot{x} + 0.6\ddot{x} + \dot{x} = x^2\left(1 - x^2\right)$ $(\ddot{x}_0, \dot{x}_0, x_0) = (0, 0, 0.4)$	
Circulant system $(Halvorsen)$ $\begin{cases} \dot{x} = 1.3x - 4y - 4z - y^2 \\ \textit{plus cyclic exchanges} \\ (x, y, z) \to (y, z, x) \to (z, x, y) \end{cases}$ $(x_0, y_0, z_0) = (-6.4, 0, 0)$	

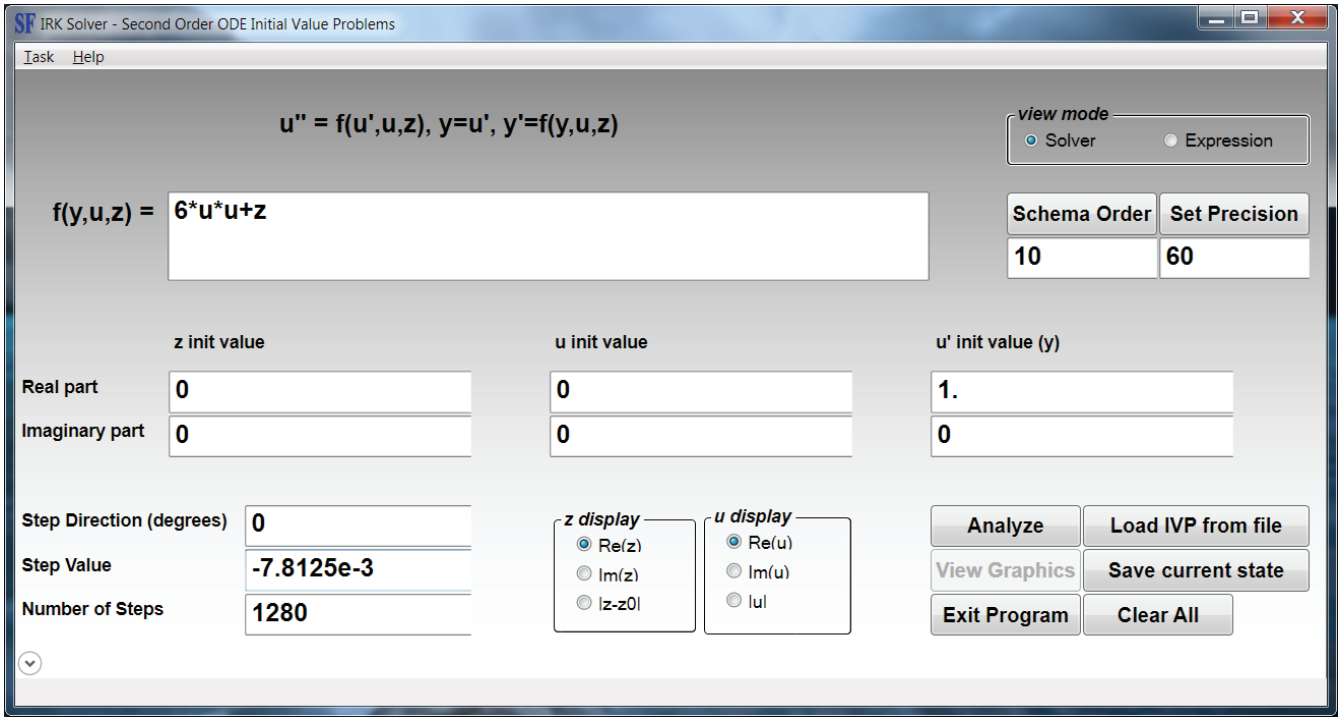


Figure 2. General view

points of the solution. The calculated points are saved in a temporary file, named sample.dat, from which gnuplot reads. This scheme is far from perfect, but has the advantage of using a free program, which can be transferred together with the other files without any installation. The names of the variables, taking part in the equation (or the functional expression) are fixed in order to avoid useless complications while formulating the problem. Besides these variables however, other variables can be present in the expression (parameters). When any parameters are present, the program requires from the user to initialize them separately. Only real constants are allowed in the main expression. If a complex constant is needed, it can be included as a parameter, to which a value is assigned with the initialization of this parameter. The basic limitation is, of course, the type of the problem to be solved (in principle the solver used is for a system of n ordinary differential equations of first order). However, even this problem (ODE of second order) in its general form is complex and interesting enough, and the idea to create this application came from the wish to explore numerically some solutions of Painlevй equations, whose properties are not yet completely studied analytically, namely because of the complexity of the problem.

In the fields for parameters initialization, a value may be entered, apriori calculated by SFCALC and stored in the clipboard.

Here an example is presented. The equation $u'' = 6u^2 + z$ (PI equation or first Painlevй equation) with 1280 steps in negative direction and a default step size, but with a negative sign and all other default initial values, gives the following result.

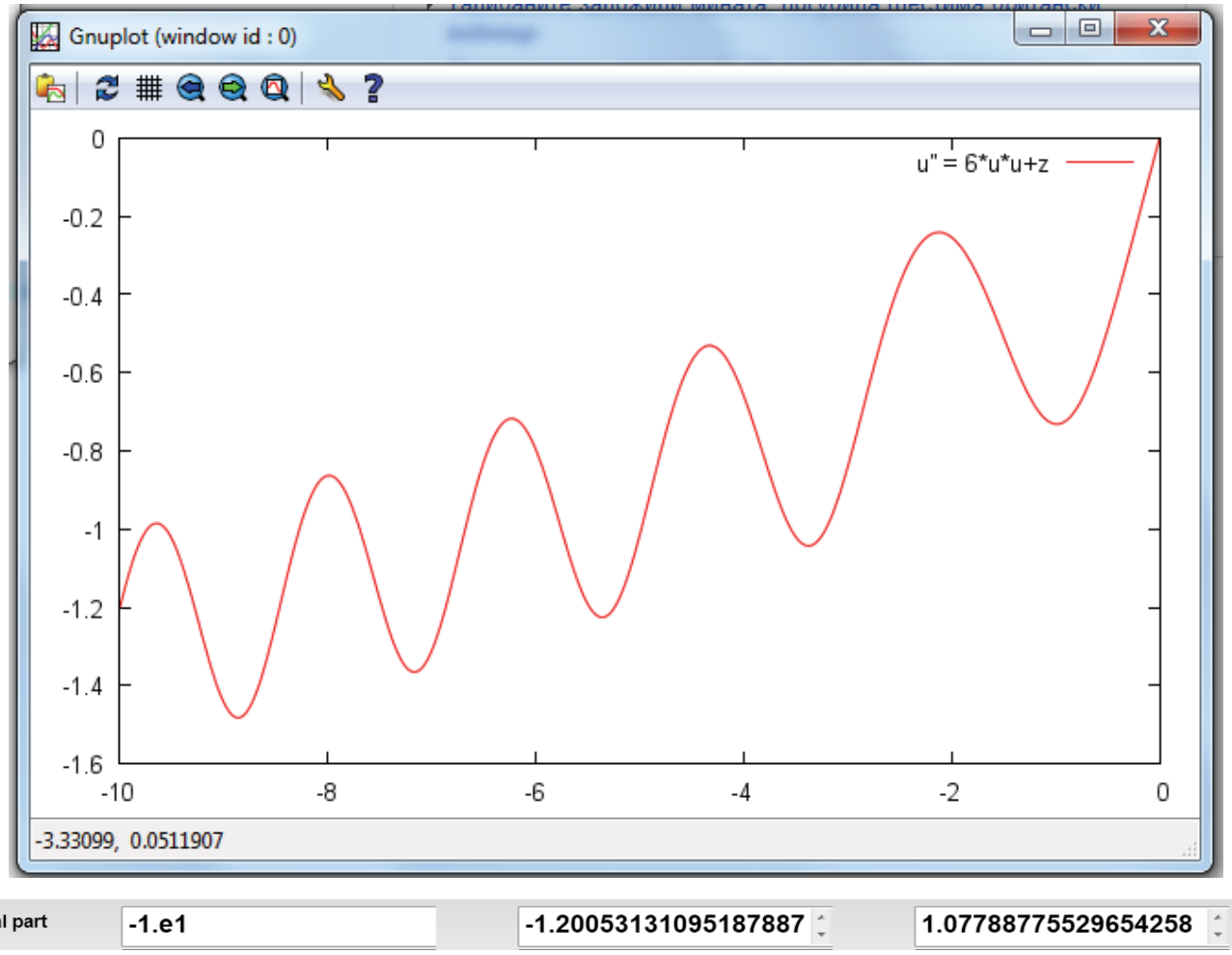


Figure 3. Solution for the given initial values

6. Conclusion

The paper presents a part of the currently done implementation of a library and tools for computing with arbitrary accuracy in environment, not typical for this purpose, namely .NET Framework. The combination of well selected methods and the excellent possibilities for integration of different functionalities in this environment are worth the efforts, and enable the achievement of non-trivial results even on a home computer. The source code will probably be proposed for free use in Internet in the near future. The purpose is to give it larger publicity to persons, interested in the development of their own tools for mathematical models so-

lution, that requires high precision and are adapted to specific research aims. The applications, herein presented, though useful, are mainly illustrations of the library possibilities and could be further improved in different aspects, including the use of the multi-core architecture of the modern processors, executing parallel calculations at some stages of the algorithms used.

Acknowledgments

The research work reported in the paper is partlially supported by the project AComIn „Advanced Computing for Innovation“, Grant 316087, funded by FP7 Capacity Programme (Research Potential of Convergence Regions).

References

1. Dekker, K., J. G. Verwer. Stability of Runge-Kutta Methods for Stiff Nonlinear Differenrial Equations. North-Holland, 1984.

2. Butcher, J. C. Numerical Methods for Ordinary Differential Equations. Second Ed. John Wiley & Sons, Ltd., 2008.

3. Hairer, E., S. P. Norsett, G. Wanner. Solving Ordinary Differential Equations I, Nonstiff Problems. Second Revised Edition, Springer, 1993.

4. http://www.unige.ch/~hairer/software.html.

5. Hairer, E., C. Lubich, G. Wanner. Geometric Numerical Integration, Structure-Preserving Algorithms for Ordinary Differential Equations. Second Edition, Springer, 2006.

6. George, E., Andrews, Richard Askey, Rnajan Roy. Special Functions. Cambridge University Press, 1999.

7. Peter, A. Clarckson. Numerics and Asymptotics for the Painlevé Equations. Numerical Solution of the Painlevй Equations. ICMS, May 2010.

8. Frank, W., J. Olver (Eds.). NIST Handbook of Mathematical Functions. National Instutute of Standards and Technology, Cambridge University Press, 2010.

9. Clarckson, Peter A. Painlevй Equations – Nonlinear Special Functions. Special Functions in the 21st Century, Washington DC, April 2011.

10. Dzhambov, V., S. Drangajov. Computing of Special Functions with Arbitrary Precision in the Environment of .NET Framework. – *Cybernetics and Information Technologies*, 11, 2011, No. 2.

11. Dzhambov, V. Using Implicit Runge-Kutta Methods for Multi Precision Solving of Nonlinear Differential Equations. – *Problems of Engineering Cybernetics and Robotics,* 62, 2010.

12. Scott, A. Sarra, Clyde Meador. On the Numerical Solution of Chaotic Dynamical Systems Using Extend Precision Floating Point Arithmetic and Very High Order Numerical Methods. – *Nonlinear Analysis: Modelling and Control,* 16, 2011, No. 3, 340-352.

13. Katok, Anatole, Boris Hasselblatt. Introduction to the Modern Theory of Dynamical Systems. Cambridge University Press, 1997.

14. Brin, Michael, Garrett Stuck. Introduction to Dynamical Systems. Cambridge University Press, 2003.

15. Robinson, Clark. Dynamical Systems. Stability, Symbolic Dynamics, and Chaos. CRC Press, 1995.

16. Wiggins, Stephen. Introduction to Nonlinear Dynamical Systems and Chaos. Second Edition, Springer, 2005.

17. Broer, Henk, Floris Takens. Dynamical Systems and Chaos. Springer, 2011.

18. Jost, Jьrgen. Dynamical Systems. Examples of Complex Behaviour. Springer, 2005.

19. Lind, Douglas, Brian Marcus. An Introduction to Symbolic Dynamics and Codding. Cambridge University Press, 1999.

20. Milnor, John. On the Concept of Attractor. – *Communications in Mathematical Physics*, 99, 1985, 177-195.

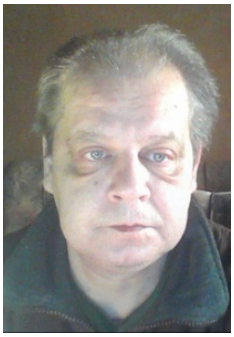
21. Milnor, John. On the Concept of Attractor: Correction and Remarks. – *Communications in Mathematical Physics,* 102, 1985, 517-519.

22. Eckmann, J.-P., D. Ruelle. Ergodic Theory of Chaos and Strange Attractors. – *Reviews of Modern Physics,* 57, July 1985, No. 3, Part I.

23. Robinson, Clark. What is Chaotic Attractor? – *Qualitative Theory of Dynamical Systems*, 7, 2008, No. 1, 227-236.

24. Martelli, Mario, Mai Dang, Tanya Seph. Defining Chaos. – *Mathematics Magazine*, 71, April 1998, No. 2.

Manuscript received on 08.05.2015



Velichko Dzhambov *was born in 1960 in Yambol, Bulgaria. M.S. degree in 1985 in physics from Sofia University "St. Kliment Ohridski". Researcher in Institute of Engineering Cybernetics and Robotics – BAS since 1988 and its later heirs till the present Institute of Information and Communication Technologies. Interests – nonlinear optimization, numerical analysis, high precision computation.*

Contacts:
Institute of Information and Communication Technologies
Bulgarian Academy of Sciences
Acad. G. Bonchev St., bl. 2, 1113 Sofia
e-mail: vili_jambov@abv.bg