GANIT *J. Bangladesh Math. Soc.* (ISSN 1606-3694) 30 (2010) 122-132

COMPARISON OF NUMERICAL METHODS FOR SOLVING INITIAL VALUE PROBLEMS FOR STIFF DIFFERENTIAL EQUATIONS

Sharaban Thohura and **Azad Rahman**

Department of Natural Science
Stamford University Bangladesh, Dhaka-1217, Bangladesh
Email: sthohura@gmail.com , azadrahman@gmail.com

Received 10.10.10 Accepted 25.04.11

ABSTRACT

Special classes of Initial value problem of differential equations termed as stiff differential equations occur naturally in a wide variety of applications including the studies of spring and damping systems, chemical kinetics, electrical circuits, and so on. Most realistic stiff systems do not have analytical solutions so that a numerical procedure must be used. In this paper we have discussed the phenomenon of stiffness and the general purpose procedures for the solution of stiff differential equation. Because of their applications in many branches of engineering and science, many algorithms have been proposed to solve such problems. In this study we have focused on some conventional methods namely Runge-Kutta method, Adaptive Stepsize Control for Runge-Kutta and an ODE Solver package, EPISODE. We describe the characteristics shared by these methods. We compare the performance and the computational effort of such methods. In order to achieve higher accuracy in the solution, the traditional numerical methods such as Euler, explicit Runge-Kutta and Adams –Moulton methods step size need to be very small. This however introduces enough round-off errors to cause instability of the solution. To overcome this problem we have used two other algorithms namely Adaptive Stepsize Control for Runge-Kutta and EPISODE. The results are compared with exact one to determine the efficiency of the above mentioned method.

1. Introduction

Many mathematical modeling give rises to systems of ordinary differential equations which although mathematically well conditioned are practically impossible to solve with traditional numerical methods because of severe step size constraint imposed by numerical stability. Mathematically these are known as stiff equations and can be characterized by the presence of transient components which are negligible relative to the other components of the solution of the problem. This characteristic of such equations constrain the step size of traditional numerical methods to be of the order of the constant of smallest component of the problem. As soon as one deals with more than one first-order differential equation, the possibility of a stiff set of equations arises. Stiffness occurs in a problem where there are two or more very different scales of the independent variable on which the dependent variables are changing. Fortunately, stiff equations generally can be predicted from the physical problem from which the equation is derived and, with care, the error can be kept under control.

Stiffness is an important concept in the numerical solution of ordinary differential equations. It depends on the differential equation, the initial conditions, and the interval under consideration. The solution of Stiff differential equations has a term of the form $\exp(-At)$, where A is a large positive constant. This is usually a part of the solution, called the transient solution. The more important portion of the solution is called the steady-state solution. The transient portion of a stiff equation rapidly decay to zero as t increases, but since the nth derivative of this term has magnitude $A^n \exp(-At)$, the derivative does not decay as quickly. In fact, since the derivative in the error term is evaluated not at t, but at a number between zero and t, the derivative terms can increase as t increase and very rapidly indeed. In case of such equations stability requirements force to take a lot of small time steps, this happens when we have a system of coupled differential equations that have two or more very different scales of the independent variable over which we are integrating. For example, suppose our solution is the combination of two exponential decay curves, one that decays away very rapidly and one that decays away very slowly. Except for the few time steps away from the initial condition, the slowly decaying curve dominate since the rapid curve will have decayed away. But because the variable time step routine to meet stability requirements for both components, we need to be confined ourselves into small time steps even though the dominant component would allow much lager time steps.

One of the first attempts to cope with the difficulties of stiffness was suggested by Curtiss and Hirschfelder (1952), who encountered stiff equations in their kinetics studies. They proposed special multistep formulas which were able to produce acceptable approximations. The difficulty of stiffness was virtually ignored by the numerical analysts until ten years later when Dahlquist (1963) identify numerical instability as the cause of the difficulty and provided some basic definitions and concepts that have been very helpful in subsequent work. Dahlquist also proposed the trapezoidal rule with extrapolation as a suitable technique to solve Stiff equations. Since Dahlquist's paper appeared, the field has been quiet active and several new approaches have been proposed for the numerical solution of stiff equations. Some of the more readily available methods for stiff equations include: Variable- order methods based on backward differentiation multistep formulas, originally analysed and implemented by Gear (1969, 1971) and later modified and studied by Hindmarsh (1974) and Byrne and Hindmarsh (1975) and Methods based on second derivative multistep formulas, such as those developed by Linger and Willoughby (1967) and Enright (1974). There are also some methods based on trapezoidal rule, such as those proposed by Dahlquist (1963) and subsequently studied by Lindberg (1971,1972) and implicit Runge-Kutta methods suitable for stiff equations, such as those based on the formulas of Butcher (1964) and studied by Ehle (1968). In addition there are some methods based on the use of preliminary mathematical transformations to remove stiffness and the solution of the transformed problem by traditional techniques, such as those studied and implemented by Lawson and Ehle (1972). Unfortunately, although a number of methods have been developed, and many more basic formulas suggested for stiff equations, until recently there has been little advice or guidance to help a practitioner choose a good method for the problem at hand.

In this paper our attention is directed towards the behavior of the solution of stiff initial value problems by two different methods. The traditional Runge-Kutta method is employed firstly and then Adaptive Stepsize Runge-Kutta method is used to get better results. Along with Runge-Kutta method an ODE solver package EPISODE is used to obtain a new set of solution. The solutions sets are compared to determine the effectiveness of these methods. The difficulties occurred in solving stiff initial value problems by the aforementioned methods have been detected and a brief discussion on handling them is given.

2. Computational Details

The specific methods that we discussed in this paper are -The fourth order Runge-Kutta method for systems, the adaptive stepsize control for Runge-Kutta and a general ODE package EPISODE. Consider the initial value problem

$$\dot{y} = \frac{dy}{dt} = f(t, y)$$

and a given initial condition, $y(t_0) = a$.

2.1 Runge-Kutta method

Runge-Kutta type method named after two German mathematicians, Runge and Kutta are called single step method because it uses only the information from the last step computed. This algorithm solves a differential equation efficiently and yet is equivalent of approximating the exact solution by matching the first n terms of the Taylor series expansion. We will consider only fourth order Runge-Kutta method, which is a higher order method. Fourth-order Runge-Kutta methods are most widely used and reliable methods for finding solutions of higher order ordinary differential equations.

2.2 Adaptive Stepsize Control for Runge-Kutta

A good ODE integrator should exert some adaptive control over its own progress, making frequent changes in its step size. Usually the purpose of this adaptive step size is to achieve some predetermined accuracy in the solution with minimum computational effort. Implementation of adaptive step size control requires that the algorithm return information about its performance, most important, an estimate of its truncation error.

2.3 Solver Package EPISODE

In a number of areas, particularly in chemical applications one often encounters systems of ordinary differential equations which, although mathematically well conditioned, are virtually impossible to solve with traditional numerical methods because of the severe step size constraint imposed by numerical stability. Over the last three decades, there has been significant progress in the development of numerical stiff ODE solvers. Consequently, a wide variety of very efficient and reliable ODE solvers have been developed such as DIFSUB, GEAR, LSODE,EPISODE, VODE, LSODPK and VODPK. The EPISODE program is a package of FORTRAN subroutines aimed at the automatic solution of problems, with a minimum effort required in the face of possible difficulties in the problem. The program implements both a generalized Adams method, well suited for nonstiff problems, and a generalized backward differentiation formula (BDF), well suited for stiff problems. Both methods are of implicit multistep type. In solving stiff problems, the package makes the heavy use of the $N \times N$ Jacobian matrix,

$$J = \frac{\partial \mathbf{f}}{\partial \mathbf{y}} = \left(\frac{\partial \mathbf{f}_i}{\partial \mathbf{y}_j} \right)_{i,j=1}^{N}$$

the $\mathbf{f}_i$ and $\mathbf{y}_j$ are the vector components of **f** and **y,** respectively.

A complete discussion of the use of EPISODE is given in [9]. However, a few basic parameter definitions are needed here, in order to present the examples. Beyond the specification of the problem itself, represented by example 1 and perhaps example 2, the most important input parameter to EPISODE is the method flag, MF. This has eight values-10, 11, 12, 13, 20, 21, 22, and 23. The first digit of MF, called METH, indicates the two basic methods to be used namely implicit Adams and BDF. The second digit, called MITER, indicates the method of iterative solution of the implicit equations arising from the chosen formula. The parameter MITER takes four different values (0, 1, 2, 3) to indicate the following respectively

- Functional (or fixed-point) iteration (no Jacobian matrix used.).
- A chord method (or generalized Newton method, or semi-stationary Newton iteration) with Jacobian given by a subroutine supplied by the user.
- A chord method with Jacobian generated internally by finite differences.
- A chord method with a diagonal approximation to the Jacobian, generated internally (at less cost in storage and computation, but with reduced effectiveness).

The EPISODE package is used by making calls to a driver subroutine, EPSODE, which in turn calls other routines in the package to solve the problem. The function **f** is communicated by way of a subroutine, DIFFUN, which the user must write. A subroutine for the Jacobian, PEDERV, must also be written. Calls to EPSODE are made repeatedly, once for each of the user's output points. A value of t at which output is desired is put in the argument TOUT to EPSODE, and when TOUT is reached, control returns to the calling program with the value of y at t =TOUT. Another argument to EPSODE, called INDEX, is used to convey whether or not the call is the first one for the problem (and thus whether to initialize various variables). It is also used as an output argument, to convey the success or failure of the package in performing the requested task. Two other input parameters. EPS and IERROR, determine the nature of the error control performed within EPISODE.

The EPISODE package consists of eight FORTRAN subroutines, to be combined with the user's calling program and Subroutines DIFFUN and PEDERV. As discussed earlier, only Subroutine EPSODE is called by the user; the others are called within the package. The functions of the eight package routines can be briefly summarized as follows:

- EPSODE sets up storage, makes calls to the core integrator, TSTEP, checks for and deals with error returns, and prints error message as needed.
- INTERP computes interpolated values of y (*t*) at the user specified output points, using an array of multistep history data.
- TSTEP performs a single step of the integration, and does the control of local error (which entails selection of the step size and order) for that step.
- COSET sets coefficients that are used by TSTEP, both for the basic integration step and for error control.
- ADJUST adjusts the history array when the order is reduced.
- PSET sets up the matrix $p = I - h\beta_0 J$, where *I* is the identity matrix, *h* is the step size, β_0 is a scalar related to the method, and J is the Jacobian matrix. It then processes P for subsequent solution of linear algebraic system with P for subsequent solution of linear algebraic systems with P as coefficient matrix.
- DEC performs an LU (lower-upper triangular) decomposition of an $N \times N$ matrix.
- SOL solves linear algebraic systems for which the matrix was factored by DEC.

The subroutine EPSODE based on variable coefficient backward differentiation formula can be used. The nonstiff option uses an Adams-Bashforth predictor and an Adams-Moulton corrector.

Predictor: $y_{n+1} = y_n + h\sum_{i=1}^{k} \beta_i y'_{n+1-i}$ & Corrector: $y_{n+1} = y_n + h\sum_{i=0}^{k} \beta_i y'_{n+1-i}$

The order may vary from one to seven.

3. Numerical Implementation and discussion

In order to compare the above mentioned methods we consider the following system of initial-value problem. This problem has been selected intentionally from Burden [2] since the exact solution is available to compare with.

$$u_1' = 9u_1 + 24u_2 + 5\cos t - \frac{1}{3}\sin t, \qquad u_1(0) = \frac{4}{3}$$

$$u_2' = -24u_1 - 51u_2 - 95\cos t + \frac{1}{3}\sin t, \qquad u_2(0) = \frac{2}{3}$$

has the unique solution

$$u_1(t) = 2e^{-3t} - e^{-39t} + \frac{1}{3}\cos t.$$

$$u_2(t) = -e^{-3t} + 2e^{-39t} - \frac{1}{3}\cos t$$

The transient term e^{-39t} in the solution causes this system to be stiff. We have solved the problem using Runge-Kutta method, Adaptive Stepsize Control for Runge-Kutta and an ODE Solver package, EPISODE and the results are shown in tabular form. We have also compared our results with the exact one.

We have considered the values of the step size h=0.1, 0.5, .01, .001 for the fixed step size Runge-Kutta method. Using the step size h=0.1 we have got disastrous result that is shown in Table 1 which misleads us from the solution curve by a large magnitude.

Table 1: Runge-Kutta method for H=0.1

t	Approximated value of $u_1(t)$	Exact value of $u_1(t)$	Approximated value of $u_2(t)$	Exact value of $u_2(t)$
0.000	0.13333334E+01	0.13333334E+01	0.66666669E+00	0.66666669E+00
0.100	-0.26451817E+01	0.17930626E+01	0.78445430E+01	-0.10320024E+01
0.200	-0.18451691E+02	0.14239024E+01	0.38876591E+02	-0.87468100E+00
0.300	-0.87473274E+02	0.11315765E+01	0.17648480E+03	-0.72499853E+00
0.400	-0.39407758E+03	0.90940857E+00	0.78936584E+03	-0.60821420E+00
0.500	-0.17600500E+04	0.73878783E+00	0.35210620E+04	-0.51565766E+00
0.600	-0.78487061E+04	0.60570961E+00	0.15698184E+05	-0.44041076E+00
0.700	-0.34990461E+05	0.49986026E+00	0.69981547E+05	-0.37740383E+00
0.800	-0.15598363E+06	0.41367146E+00	0.31196775E+06	-0.32295352E+00
0.900	-0.69535119E+06	0.34161434E+00	0.13907029E+07	-0.27440882E+00
1.000	-0.30997643E+07	0.27967489E+00	0.61995290E+07	-0.22988783E+00

In order to obtain better approximation from the previous one we have reduced the step size by half i.e. we have considered h=.05 and then we have observed that the results (shown in the following table) improve radically which is correct up to three significant digits.

Table 2: Runge-Kutta method for H=0.05

t	Approximated value of $u_1(t)$	Exact value of $u_1(t)$	Approximated value of $u_2(t)$	Exact value of $u_2(t)$
0.000	0.13333334E+01	0.13333334E+01	0.66666669E+00	0.66666669E+00
0.100	0.17122205E+01	0.17930626E+01	-0.87031507E+00	-0.10320024E+01
0.200	0.14140718E+01	0.14239024E+01	-0.85501510E+00	-0.87468100E+00
0.300	0.11305257E+01	0.11315765E+01	-0.72289181E+00	-0.72499853E+00
0.400	0.90927821E+00	0.90940857E+00	-0.60794848E+00	-0.60821420E+00
0.500	0.73875201E+00	0.73878783E+00	-0.51558137E+00	-0.51565766E+00
0.600	0.60568446E+00	0.60570961E+00	-0.44035622E+00	-0.44041076E+00
0.700	0.49983734E+00	0.49986026E+00	-0.37735450E+00	-0.37740383E+00
0.800	0.41365030E+00	0.41367146E+00	-0.32290822E+00	-0.32295352E+00
0.900	0.34159508E+00	0.34161434E+00	-0.27436787E+00	-0.27440882E+00
1.000	0.27965778E+00	0.27967489E+00	-0.22985160E+00	-0.22988783E+00

To improve our result we again attempt with a new step size (h=.01) which is very small compared to the earlier values and with this change the result agrees with the exact value up to 5 significant figures.

Table 3: Runge-Kutta method for H=0.01

t	Approximated value of $u_1(t)$	Exact value of $u_1(t)$	Approximated value of $u_2(t)$	Exact value of $u_2(t)$
0.000	0.13333330E+01	0.13333334E+01	0.66666669E+00	0.66666669E+00
0.100	0.17930413E+01	0.17930626E+01	-0.10319601E+01	-0.10320024E+01
0.200	0.14239013E+01	0.14239024E+01	-0.87467921E+00	-0.87468106E+00
0.300	0.11315763E+01	0.11315765E+01	-0.72499835E+00	-0.72499859E+00
0.400	0.90940845E+00	0.90940863E+00	-0.60821414E+00	-0.60821420E+00
0.500	0.73878777E+00	0.73878783E+00	-0.51565760E+00	-0.51565766E+00
0.600	0.60570961E+00	0.60570967E+00	-0.44041070E+00	-0.44041076E+00
0.700	0.49986029E+00	0.49986026E+00	-0.37740383E+00	-0.37740383E+00
0.800	0.41367149E+00	0.41367149E+00	-0.32295352E+00	-0.32295352E+00
0.900	0.34161437E+00	0.34161437E+00	-0.27440882E+00	-0.27440885E+00
1.000	0.27967489E+00	0.27967492E+00	-0.22988783E+00	-0.22988784E+00

Again, reducing the value of h (h=.001) we have observed that our result shows a very good conforms to the exact solution (up to 7 digits) of the problem.

Table 4: Runge-Kutta method for H=0.001

t	Approximated value of $u_1(t)$	Exact value of $u_1(t)$	Approximated value of $u_2(t)$	Exact value of $u_2(t)$
0.000	0.13333330E+01	0.13333334E+01	0.66666669E+00	0.66666669E+00
0.100	0.17930620E+01	0.17930626E+01	-0.10320022E+01	-0.10320024E+01
0.200	0.14239023E+01	0.14239024E+01	-0.87468088E+00	-0.87468100E+00
0.300	0.11315769E+01	0.11315765E+01	-0.72499871E+00	-0.72499853E+00
0.400	0.90940911E+00	0.90940857E+00	-0.60821444E+00	-0.60821420E+00
0.500	0.73878819E+00	0.73878783E+00	-0.51565784E+00	-0.51565766E+00
0.600	0.60570997E+00	0.60570961E+00	-0.44041094E+00	-0.44041073E+00
0.700	0.49986023E+00	0.49986023E+00	-0.37740383E+00	-0.37740380E+00
0.800	0.41367146E+00	0.41367143E+00	-0.32295352E+00	-0.32295349E+00
0.900	0.34161443E+00	0.34161431E+00	-0.27440885E+00	-0.27440882E+00
1.000	0.27967495E+00	0.27967489E+00	-0.22988784E+00	-0.22988781E+00

In case of fixed step size Runge-Kutta method we have to solve the problem with a predetermined step size and so there is a possibility of getting the incorrect result. As a result we are also not able to achieve our desired accuracy. In this case we have tried with more than one step size and proceeds up to last step which is time consuming. To overcome this problem we have solved this problem with Adaptive Step size control for Runge-kutta. In this method variable step sizes are used in each step to achieve a predefined accuracy. The following table shows the result of the problem using Adaptive Stepsize control for Runge-Kutta.

Table 5: Adaptive Stepsize control for Runge-Kutta

t	Approximated value of $u_1(t)$	Exact value of $u_1(t)$	Approximated value of $u_2(t)$	Exact value of $u_2(t)$
0.000	0.13333330E+01	0.13333334E+01	0.66666669E+00	0.66666669E+00
0.100	1.79304858406938	0.17930626E+01	-1.03197450514632	-0.10320024E+01
0.200	1.42390207492145	0.14239024E+01	-0.874680422284331	-0.87468100E+00
0.300	1.13157649536409	0.11315765E+01	-0.724998544910728	-0.72499853E+00
0.400	0.909408571598146	0.90940857E+00	-0.608214198568701	-0.60821420E+00
0.500	0.738787825268748	0.73878783E+00	-0.515657666707487	-0.51565766E+00
0.600	0.605709637966968	0.60570961E+00	-0.440410753275052	-0.44041073E+00
0.700	0.499860243643682	0.49986023E+00	-0.377403817988363	-0.37740380E+00
0.800	0.413671468596188	0.41367143E+00	-0.322953517215125	-0.32295349E+00
0.900	0.341614340930095	0.34161431E+00	-0.274408829650564	-0.27440882E+00
1.000	0.279674898219440	0.27967489E+00	-0.229887831037646	-0.22988781E+00

Since most real life Stiff initial value problems cannot be solved with the traditional numerical methods because of the severe step size constraint imposed by numerical stability and so a number of very efficient ODE solvers have been developed. To justify our result we have solved our problem with a general ODE solver package EPISODE and the following table summarizes the result.

Table 6: Episode:

t	H	Approximated Value of $u_1(t)$	Exact value of $u_1(t)$	Approximated Value of $u_2(t)$	Exact value of $u_2(t)$
0.0	.40E-02	1.33333333	1.33333333	0.666666666	0.666666666
0 .1	.40E-02	1.79306146	1.79306300	-1.03200020	-1.03200200
0.2	.91E-02	1.42390205	1.42390200	-0.87468033	-0.87468100
0.3	.12E-01	1.13157624	1.13157700	-0.72499799	-0.72499860
0.4	.33E-01	0.90940824	0.90940860	-0.60821345	-0.60821420
0.5	.33E-01	0.73878794	0.73878780	-0.51565752	-0.51565770
0.6	.45E-01	0.60571002	0.60570970	-0.44041090	-0.44041080
0.7	.66E-01	0.49986115	0.49986030	-0.37740429	-0.37740380
0.8	.66E-01	0.41367419	0.41367150	-0.32295478	-0.32295350
0.9	.66E-01	0.34161891	0.34161430	-0.27441118	-0.27440880
1.0	.66E-01	0.27968063	0.27967490	-0.22989065	-0.22988780

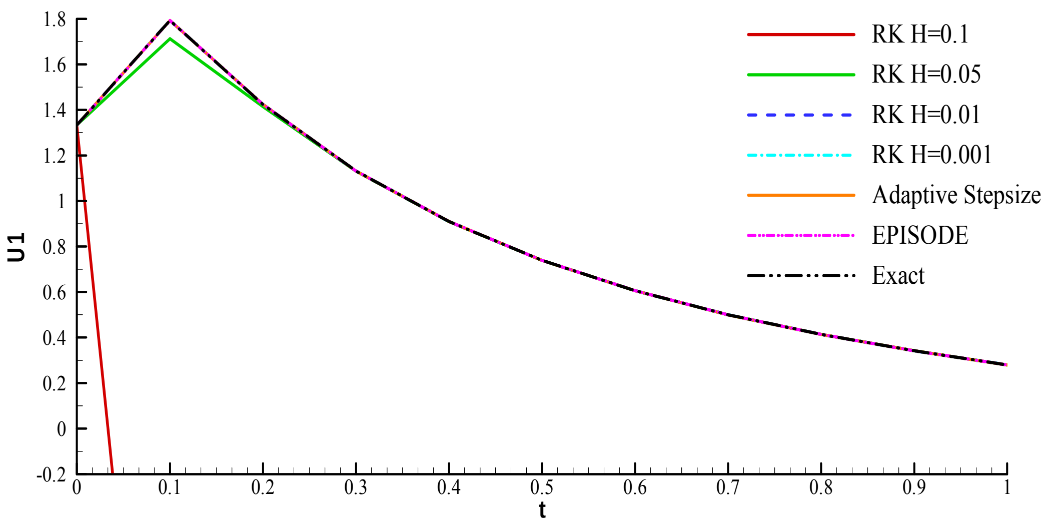

Figure 1: The graph of the solutions for u $_1$

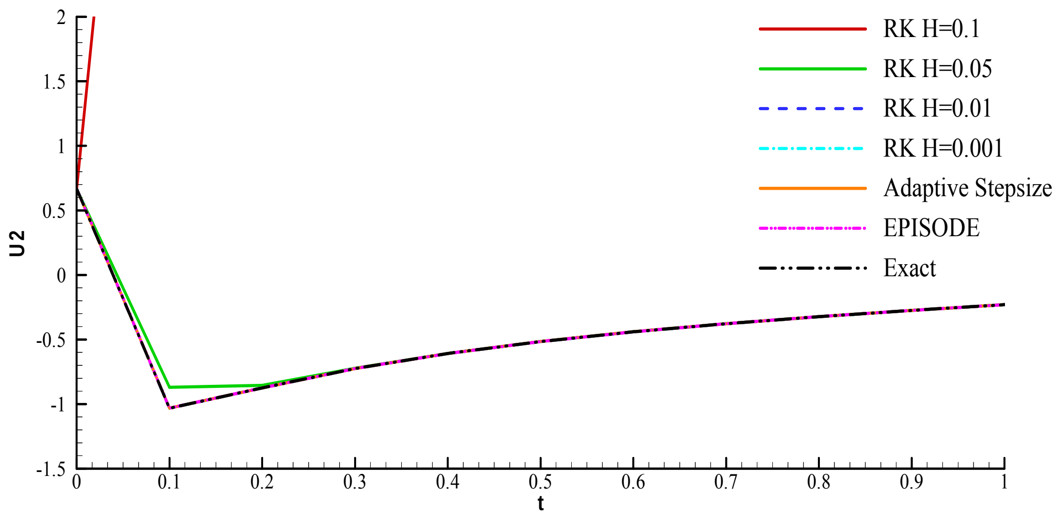

Figure 2: The graph of the solutions for u_2

From the figures (1) and (2) we observed that except Runge-Kutta method with H = 0.1 and Runge-Kutta method with H = 0.05 all other solutions agree with the exact one.

Conclusion

There are many basic formulas and effective codes available to solve such problems, but until now there has a little advice or guidance to help a user to choose a good method for solving such stiff initial value problem. In this paper we focus on Runge-Kutta method, Adaptive Stepsize Control for Runge-Kutta and an ODE Solver package, EPISODE. Important finding throughout the discussion is that the fixed step size Runge-Kutta method is inappropriate for stiff differential equations. For the fixed step size Runge-Kutta method, we observe that with decreasing the step size the accuracy of the approximation improves. Still the user faces the problem that how small the step size should be to obtain the desired accuracy. On the other hand the Adaptive step size Runge-Kutta method allows us to set the accuracy level where step size changes for different values of the independent variable. In contrast EPISODE is a solver package which is much more user friendly and provides a very good approximation though the accuracy level is not as good as Adaptive step size Runge-Kutta method. But EPISODE is more effective for real life problem where the system is too large and complicated.

REFERENCES

1. Atkinson, Kendall E., An introduction to Numerical Analysis, John Wiley & Sons, 1989.
2. Burden, R. L.and Faires, J. D. Numerical Analysis, PWS Publishers, U.S.A., 2002.
3. Byrne, George D and Hindmarsh, A.C. Stiff ODE Solvers: A review of current and coming attractions. Journal of Computational Physics. 70(1) (1986), 1-62.

4. Byrne, G. D. and Hindmarsh, A. C., A polyalgorithm for the numerical solution of ordinary differential equations. ACM Transactions on Mathematical Software, 1(1975), 71-96.

5. Enright, W.H. and HULL, T.E. Comparing Numerical Methods for the solution of Stiff Systems of ODEs Arising in Chemistry, Numerical Method, Academic Press Orlando, 1976.

6. Gear, C.W., The automatic integration of stiff ordinary differential equation. Information Processing 68 (1969), A. J. H. Morrell, Ed., North Holland, Amsterdam, 187-193.

7. Gear, C.W. Algorithm 407: DIFSUB for solution of ordinary differential equations. Comm. ACM, 14 (1971), 185-190.

8. Gerald, F. Curtis and Wheatley O. Patrick. Applied Numerical Analysis, Pearson Education, Inc, 1999.

9. Hindmarsh, A. C. and Byrne, G. D. Episode. An experimental package for the integration of systems of ordinary differential equations, L.L.L. Report UCID-30112, l.l.l., (1975) (www.netlib.org/ode/epsode.f)

10. Hindmarsh, A. C. and Gear C.W., Ordinary differential equation system solver, L.L.L. Report UCID - 30001, rev. 3(1974), l.l.l. (www.netlib.org/ode/epsode.f)

11. Lapidus, L. and Schiesser, W.E., Numerical Methods for Differential System, Academic press, INC. (London), 1976.

12. Nejad, Lida A.M.(2005) A comparison of stiff ODE solvers for astrochemical kinetics problems, Astrophysics and Space Science. 299(2005), 1-29.

13. Shampine, L. F. and Gear, C.W., A user's View of Solving Stiff Ordinary Differential Equations, SIAM Review, Vol.21(1979), 1-17.