Journal of Scientific Research & Reports
3(11): 1430-1448, 2014; Article no. JSRR.2014.11.002

SCIENCEDOMAIN *international*
www.sciencedomain.org

Identifying Stiff Ordinary Differential Equations and Problem Solving Environments (PSEs)

B. K. Aliyu[1*], C. A. Osheku[1], A. A. Funmilayo[1] and J. I. Musa[1]

[1]Federal Ministry of Science and Technology (FMST), National Space Research and Development Agency (NASRDA), Centre For Space Transport and Propulsion (CSTP) Epe, Lagos State, Nigeria.

Authors' contributions

This work was carried out in collaboration between all authors. Author BKA designed the study, carried out the simulations and wrote the first draft of the manuscript. Author CAO supervised the study and re-drafted the manuscript. Authors AAF and JIM verified all simulations results and literature search. All authors read and approved the final manuscript.

Original Research Article

Received 3rd March 2014
Accepted 4th April 2014
Published 16th April 2014

ABSTRACT

Stiff Ordinary Differential Equations (SODEs) are present in engineering, mathematics, and sciences. Identifying them for effective simulation (or prediction) and perhaps hardware implementation in aerospace control systems is imperative. This paper considers only linear Initial Value Problems (IVPs) and brings to light the fact that stiffness ratio or coefficient of a suspected stiff dynamic system can be elusive as regards the phenomenon of stiffness. Though, it gives the insight suggesting stiffness when the value is up to 1000 but is not necessarily so in all ODEs. Neither does a value less than 1000 imply non-stiffness. MATLAB/Simulink® and MAPLE® were selected as the Problem Solving Environment (PSE) largely due to the peculiar attribute of Model Based Software Engineering (MBSE) and analytical computational superiority of each PSEs, respectively. This creates the base for comparing results from both numerical and analytical standpoint. In Simulink, two methods of modelling ODEs are presented. Experimenting with all the variable-step solvers in MATLAB® ODE Suit for selected examples was carried out. Results point to the fact that stiffness coefficient of about 1000does not always suggest that an ODE is stiff nor does a value less than 1000 suggest non-stiff.

Keywords: ODEs; stiffness ratio; MATLAB/Simulink®; MAPLE®; variable-step solvers.

**Corresponding author: E-mail: aliyu_bhar@yahoo.com;*

1. INTRODUCTION

An ordinary linear differential equation is known to model any linear system, relating the output response of the system to the input, whether electrical, mechanical, hydraulic, or thermal. The analogies between these several engineering disciplines have been developed over time [1]. ODE problem has been divided into stiff and non-stiff problems. In mathematics, a stiff equation is a differential equation for which certain numerical methods for solving the equation are numerically unstable, unless the step size is taken to be extremely small. It has proved difficult to formulate a precise definition of stiffness, but the main idea is that the equation includes some terms that can lead to rapid variation in the solution. Stiffness is a subtle, difficult and important concept in the numerical solution of ordinary differential equations. It depends on the differential equation, the initial conditions, and the numerical method.

The best way to detect stiffness is to try one of the solvers intended for non-stiff systems. If it is unsatisfactory, the problem maybe stiff. If the problem is stiff, there are effective solvers available [2].

The phenomenon of stiffness is not precisely defined in literature. Some attempts at describing a stiff problem are:

- A differential equation of the form $y' = f(t,y(t))$ is said to be stiff if its exact solution $y(t)$ includes a term that decays exponentially to zero as t increases, but whose derivatives are much greater in magnitude than the term itself. An example of such a term is $e^{-\lambda t}$, where λ is a large, positive constant, because its kth derivative is $c^k e^{-\lambda t}$. Because of the factor of c^k, this derivative decays to zero much more slowly than $e^{-\lambda t}$ as t increases. Because the error includes a term of this form, evaluated at a time less than t, the error can be quite large if h which is the step size is not chosen sufficiently small to offset this large derivative. Furthermore, the larger λ is, the smaller h must be to maintain accuracy.
- A problem is stiff if it contains widely varying time scales, i.e., some components of the solution decay much more rapidly than others,
- A problem is stiff if the step size is dictated by stability requirements rather than by accuracy requirements.
- A problem is stiff if explicit methods don't work, or work only extremely slowly.
- A linear problem is stiff if all of its eigenvalues have negative real part, and the stiffness ratio (the ratio of the magnitudes of the real parts of the largest and smallest eigenvalues) is large.

It is pertinent to note that in this research we are concerned with the computational aspect of the properties that define stiffness. If we weren't concerned with how many computational steps an algorithm takes to converge to a solution which implies more storage space on a microcontroller for control system hardware implementation, we wouldn't be concerned about stiffness. Non-stiff methods can solve stiff problems; they just take a long time to do it. Also, more computational steps translate into longer time for an algorithm to converge to a solution. Both issues are crux in control system design.

2. STIFFNESS RATIO OR COEFFICIENT

The following definitions are popular as regards a numerical factor attributed to the degree of stiffness:

Definition 1.1 considering a higher order system defined as

$$y' = Ay \; y(t_0) = y_0, \tag{1}$$

with an $n \times n$ matrix A having eigenvalues $\lambda_i (i=1,\ldots,n)$ which all have a negative real part. If these real parts differ considerably, the initial value problem is called *stiff*. A measure for the stiffness is the so called stiffness coefficient, expressed as

$$S_1 := \frac{\max_i |\Re(\lambda_i)|}{\min_i |\Re(\lambda_i)|} \tag{2}$$

Note that both λ_{max} and λ_{min} must be greater than zero for stiffness to occur. If the coefficients of the ODE (i.e., the matrix elements) comprise several orders of magnitude, S can reach values of $\vartheta(10^6)$ or even more.

Definition 1.2 generally, a dimensionless index of stiffness for every system is given as

$$S_2 = \tau(b-a), \tag{3}$$

Where *[a, b]* is the integration interval of interest, $\tau = |\mathrm{Re}(\lambda_i)|$, and λ_i is the eigenvalue of the largest negative real part. Actually τ varies along the solution.

While the intuitive meaning of stiff is clear to all specialists, much controversy is going on about its correct mathematical definition. The most pragmatic opinion is also historically the first one which states that *Stiff* equations are equations where certain implicit methods, in particular backward differentiation methods, perform much better than explicit ones. Stiff ODEs defeat the explicit *Runge-Kutta* and *Adams-Bashforth* methods. In fact, for such problems, the higher order methods perform even more poorly than the low order methods [3].

2.1 Stiffness

The common approach to understanding stiffness is motivated by the behaviour of fixed step size solutions for systems of linear ODEs with constant coefficients. The eigenvalues of the coefficient matrix completely characterize the solution of the system and they determine the behaviour of numerical methods applied to the system. Generally, stiffness is considered tobe important for systems in which all of the eigenvalues λ_i of the coefficient matrix have negative real parts, that is, to systems that are mathematically stable.

When explicit methods such as those in *ode45*are used, stiffness imposes restrictions on the step size required to obtain a stable numerical approximation. It is easy to see how such restrictions arise by considering what happens when the Forward Euler method given in (4),

with h as the step size and $t_{n+1}=t_n+h$ is a sequence of time at which the approximate solution $y_{n+1}$is desired

$$y_{n+1} = y_n + hf\left(t, y_n\right), \tag{4}$$

is applied to the linear test equation given in (5).A simple calculation shows that the numerical solution is given by (6)

$$y'\left(t\right) = \lambda_i y\left(t\right),\ y\left(0\right) = 1. \tag{5}$$

$$y_n = \left(1 + h\lambda_i\right)^n. \tag{6}$$

The numerical solution decays to 0 as t_n increases only if $|(1 + h\lambda_i)| < 1$.The complex values for which this is the case form the *interior* of a unit circle cantered at the point (-1,0). Since this restriction must hold for each of the eigenvalues, it imposes a severe restriction on the step size when $S$is large. Any explicit method has a finite stability region determined by the above condition.

By way of contrast, some implicit methods have infinite stability regions. A first example is the Backward Euler method given in (7) which generates the approximations given by (8).

$$y_{n+1} = y_n + hf\left(t, y_{n+1}\right). \tag{7}$$

$$y_n = \frac{1}{\left(1 - h\lambda_i\right)^n}. \tag{8}$$

These approximations decay to 0 for points *exterior* to the unit circle centred at the point (1,0). In particular, the resulting stability region contains the entire left half-plane. If the ODE system is stable, no numerical stability restriction is thus placed on the step size which is then controlled by keeping local errors sufficiently small. The MATLAB stiff ODE solver *ode15s* implements this method and similar higher order 'stiffly stable' methods whose stability regions contain a significant portion of the left half-plane. Imagine that an explicit method such as *ode45* is being used. Step sizes will be used for which hS is near the stability boundary for the method and for some step sizes hS will fall outside the stability region. Thereupon is where the stability of the ODE system comes into play.

In the event a system is stable, we can obtain an indication of stiffness by the size of $S(b-a)$. An excessive amount of work may be required even if S is of moderate size if $b-a$ is large enough. Similarly, even if S is very large, an excessive amount of work is not necessarily required if $b-a$ is small enough. In fact, the size of S $(b-a)$ is what determines the amount of work required and is therefore the commonly accepted indicator of stiffness [4].

This paper examines 6 simple examples with different values of stiffness ratio; all solutions to the selected problems will require the solution of a differential equation; a method using the symbolic processing capabilities of MAPLE® to quickly code a differential equation to obtainits closed-form solution and the numerical solution will stern from a MATLAB/Simulink® model. Both PSEs are most equipped for such task .The analytical solution serves as the

benchmark for comparison of results. Also, all MATLAB® ODE variable-step *solvers* will be experimented on each problem.

3. PROBLEM SOLVING ENVIRONMENTS (PSEs)

The Problem Solving Environments (PSEs) Maple® and MATLAB® are in very wide use. Although they have much in common, they are clearly distinguished by the emphasis in Maple on algebraic or symbolic computation and MATLAB on numerical computation [5].

4. SIMULATIONS

It is worth mentioning here that MATLAB® 2012b and MAPLE® 16 were used for all the computations in this research on a computer with the following configuration: *Processor*- Intel(R) Core(TM) i5-2430M CPU @ 2.40GHz, *RAM*- 4.00GB. For all the simulations in Simulink®, *RelTol*=0.001 and *AbsTol*=10^{-6}. In Simulink, Model I is built base on the traditional method of modelling using pre-defined blocks while Model II uses the special block of *MATLAB Function* with additions from pre-defined blocks to complete each model.

Example I. We consider the system given in (9), a first order ODE [6]. In Simulink, (9) can be modelled firstly, as shown in Fig. 1.

$$\frac{dy}{dt} = -1000y + 3000 - 2000e^{-t}, \qquad y(0) = 0. \tag{9}$$

Stiffness for (9) can be considered as S_2=5000. After simulating in MATLAB/Simulink, using two different methods of building such models (Figs. 1 and 2), Table 1 below gives the result of the number of steps various solvers took to converge to a solution.

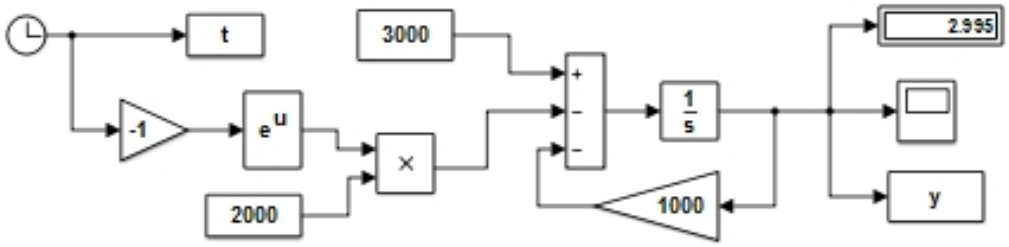

Fig. 1. Model I, Simulink model for example I

Alternatively, the MATLAB embedded function block for automatic code generation could be used to model the same system. This model is built also in the Simulink environment and is as shown in Fig. 2.

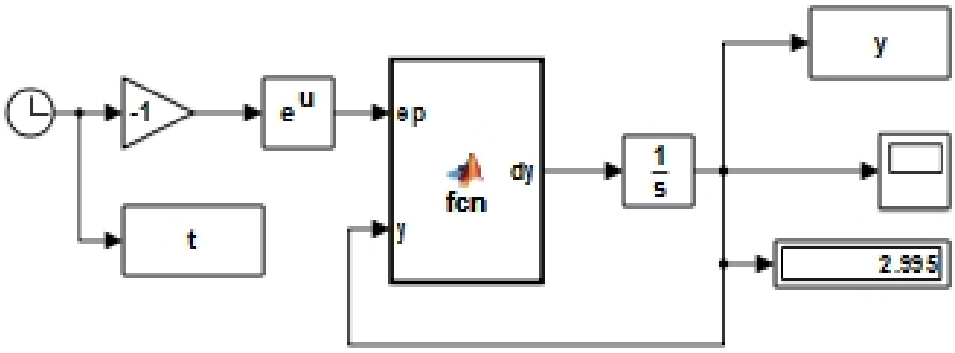

Fig. 2. Mode l l, Simulink model for example l

Experimenting with all the variable step solvers in the MATLAB ODE suit gave the results in Table 1.

Its exact solution was computed with the help of the following MAPLE® code:

restart;
with (DEtools):
*sys:= diff(y(t),t)=-1000*y(t)+3000-2000*exp(-t);*
dsolve({sys,y(0)=0,y(t)});

and its closed-form solution as given in (10). Fig. 3, gives the graphical solution.

$$y(t) = 3 - 2.002e^{-t} - 0.998e^{-1000t}, \tag{10}$$

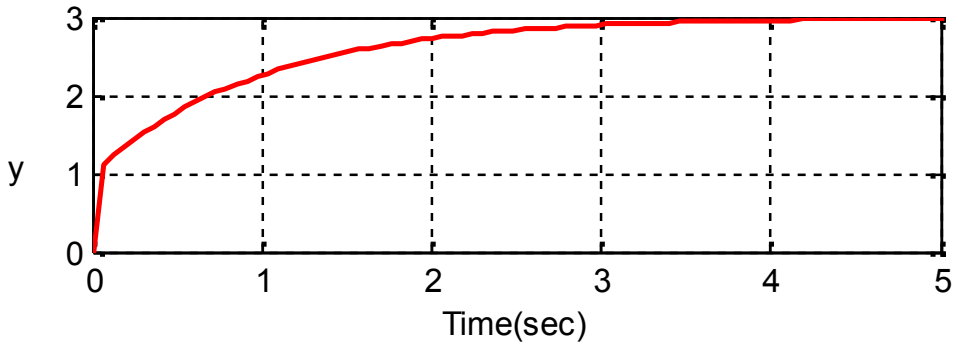

Fig. 3. Simulation result for example l

Example II. For the following system which has been considered by *Lambert* [7]:

$$y' = Ay \; y(0) = \begin{pmatrix} 1 & 0 & -1 \end{pmatrix}^T,$$

where

$$A\begin{bmatrix} -21 & 19 & -20 \\ 19 & -21 & 20 \\ 40 & -40 & -40 \end{bmatrix} \tag{11}$$

With S_1=20 and S_2=60. In Simulink, (11) is first modelled as shown in Fig. 4.

An alternative to modelling Fig. 4 is as given in Fig. 5.

Experimenting with solvers, Table 2 gives the resulting number of computational steps for solution to converge.

The following MAPLE code were used to realize the analytical solution to (11):

```
restart;
PDEtools[declear]((x,y,z,f,g)(t),prime=t);
sys:=[diff(x(t),t)=-21*x(t) + 19*y(t)-20*z(t),
diff(y(t),t)=19*x(t)-21*y(t)+20*z(t),
diff(z(t),t)=40*x(t)-40*y(t)-40*z(t), x(0)=1, y(0)=0, z(0)=-1];
sol:=dsolve(sys);
odetest(sol,sys);
```

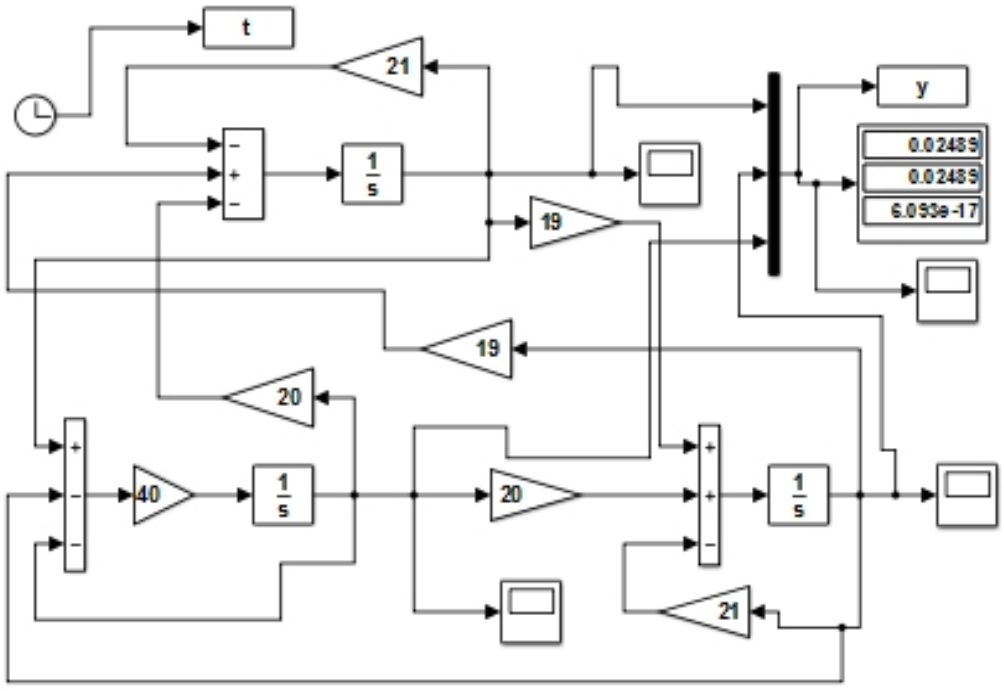

Fig. 4. Model I, Simulink model for example II

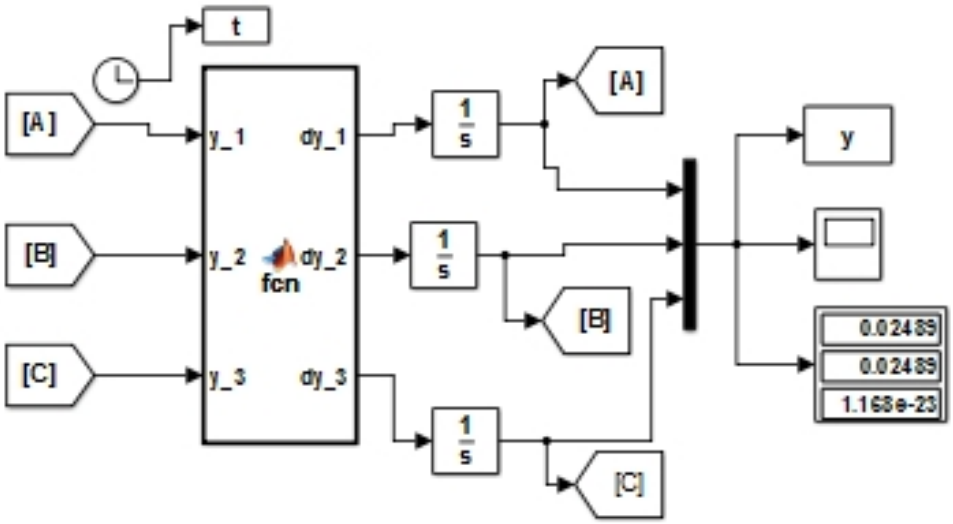

Fig. 5. Model II, Simulink model for Example II

The solutions obtained are as given in (12). Note that the state variables were changed in the MAPLE®syntax. This is necessary for easy recognition and evaluation by the software. Graphically the results are depictedin Fig. 6.

$$\begin{aligned} y_1 &= 0.5e^{-2t} + 0.5e^{-40t}\left(\cos(40t) + \sin(40t)\right), \\ y_2 &= 0.5e^{-2t} - 0.5e^{-40t}\left(\cos(40t) + \sin(40t)\right), \\ y_3 &= -e^{-40t}\left(\cos(40t) - \sin(40t)\right). \end{aligned} \tag{12}$$

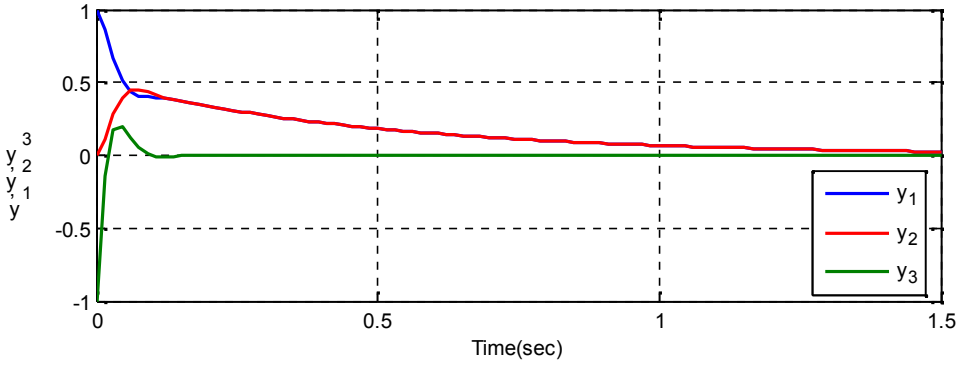

Fig. 6. Result of simulating example II

Example III. The system of equations in (13), again from Stephen C Chapra 2012 is considered.

$$\begin{aligned} \frac{dy_1}{dt} &= -5y_1 + 3y_2, \\ \frac{dy_2}{dt} &= 100y_1 - 301y_{2,} \end{aligned} \qquad y_0(0) = \begin{pmatrix} 52.29 & 83.82 \end{pmatrix}^T. \tag{13}$$

This has S_1= 101 and for t= [0 1], S_2=302.Simulink model for (13) is as shown in Fig. 7.

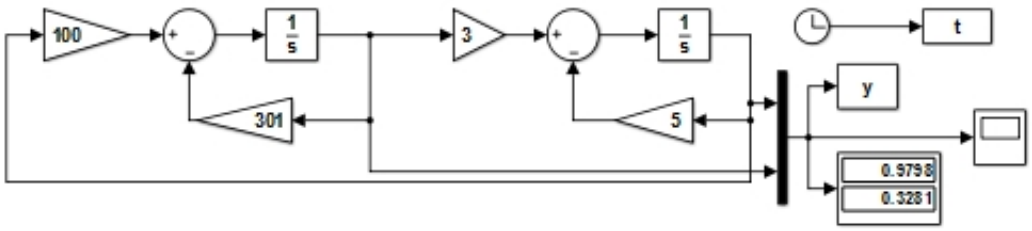

Fig. 7. Model I, Simulink model of example III

Using MATLAB Function block, the Simulink model for Example III is as shown in Fig. 7.

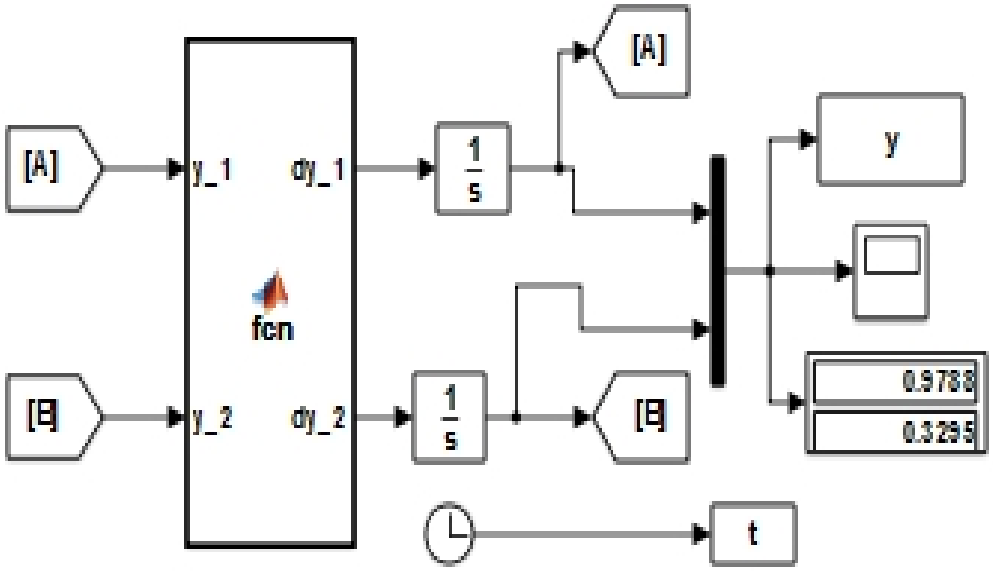

Fig. 7. Model II, Simulink model for Example III

Also experimenting with solvers, Table 3 gives the resulting steps each solver took to converge to a solution.

The MAPLE code used to obtain the closed form solution to (13)is:

```
restart;
PDEtools[declear]((x,y,z,f,g)(t),prime=t);
sys:=[diff(x(t),t)=-5*x(t) + 3*y(t), diff(y(t),t)=100*x(t)-301*y(t),
x(0)=52.29, y(0)=83.82];
sol:=dsolve(sys);
odetest(sol,sys);
```

The exact solution is given in (14) and the graphical result is shown in Fig. 8.

$$\begin{aligned} y_1 &= 52.96e^{-3.989t} - 0.67e^{-302.0101t} \\ y_2 &= 17.83e^{-3.989t} + 65e^{-302.0101t}. \end{aligned} \tag{14}$$

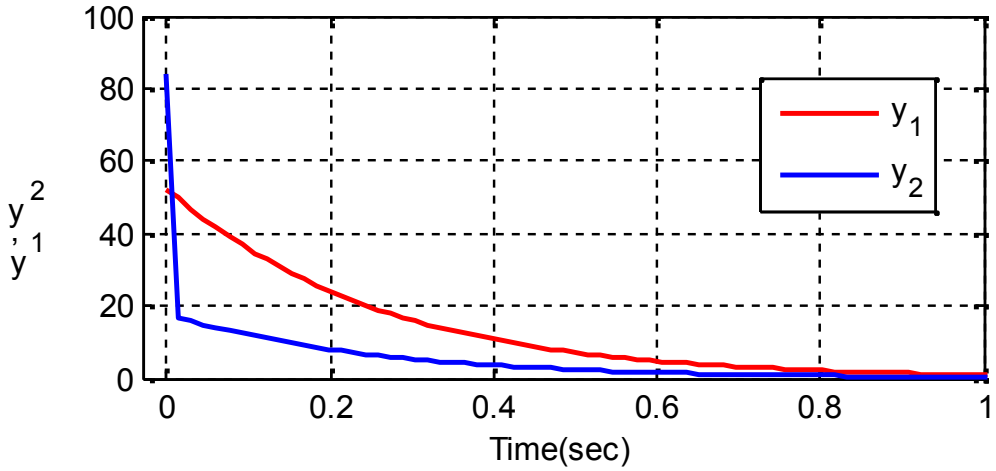

Fig. 8. Simulation result for example III

Example IV. For a system given as [8]:

$$\begin{aligned} &y_1' + 0.5y_1 = 0, \\ &y_2' + y_2 = 0, \\ &y_3' + 100y_3 = 0, \\ &y_4' + 90y_4 = 0. \end{aligned} \qquad y_0 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T \tag{15}$$

Stiffness for (15) is given as S_1=200, and S_2=1200 for t= [0 12]. Its Simulink model is shown in Fig. 9.

Alternative to Fig. 9 is shown in Fig. 10. Using the MATLAB Function block as the primary block and directly writing the equations into the block before adding the *integrator* blocks outside it.

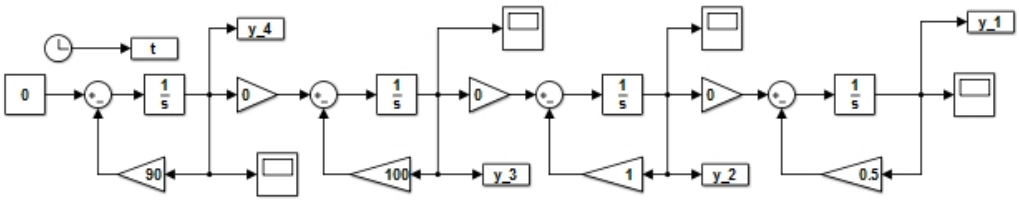

Fig. 9. Model I, Simulink mode for example IV

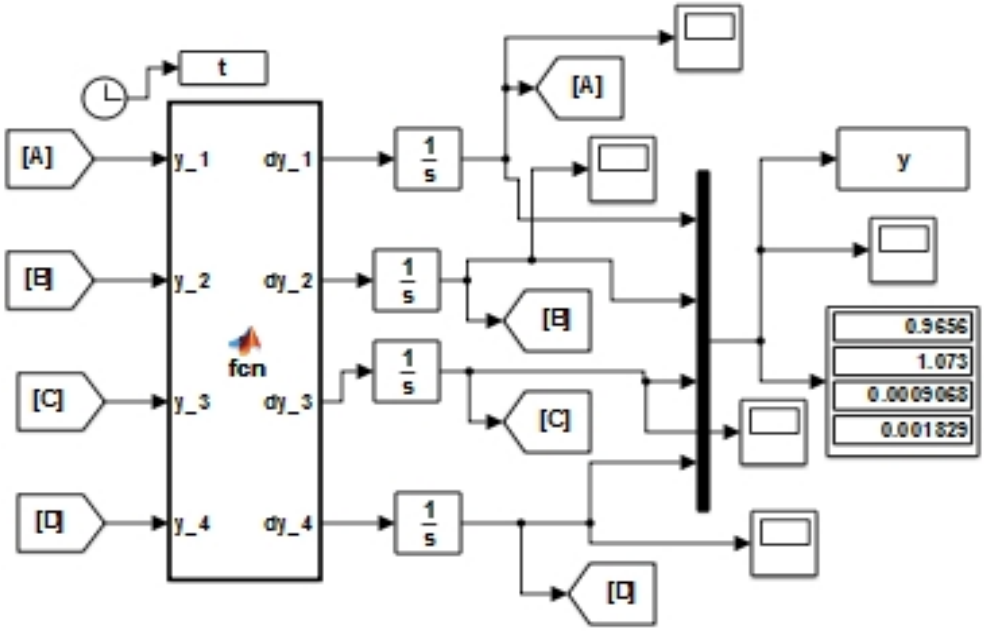

Fig. 10. Model II, Simulink model for example IV

After experimenting with different solvers, Table 4 gives the resulting number of step taken by each for solution to converge. Graphical results are shown in Fig. 11 for all the state variables.

In MAPLE® the following codes were used for the closed form solution with result in (16):

```
restart;
PDEtools[declear]((x,y,z,f,g)(t),prime=t);
sys:=[diff(x(t),t)=-0.5*x(t),diff(y(t),t)=-y(t),
diff(z(t),t)=-100*z,diff(z(t),t)=-90*f(t), x(0)=1, y(0)=1, z(0)=1, f(0)=1];
sol:=dsolve(sys);
odetest(sol,sys);
```

$$
\begin{aligned}
y_1 &= e^{-0.5t}, \\
y_2 &= e^{-t}, \\
y_3 &= e^{-100t}, \\
y_4 &= e^{-90t}.
\end{aligned}
\tag{16}
$$

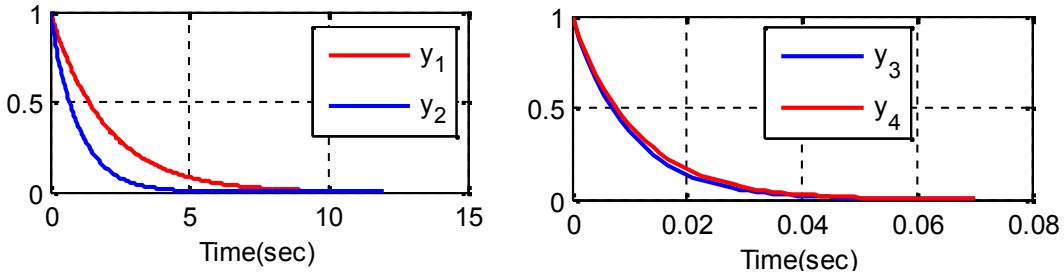

Fig. 11. Simulation result for example IV

Example V. The system described by (8), is considered [9].

$$
\begin{aligned}
&y_1' + 0.1y_1 + 49.9y_2 = 0, \\
&y_2' + 50y_2 = 0, \qquad y_o(0) = \begin{pmatrix} 2 & 1 & 2 \end{pmatrix}^T. \\
&y_3' - 70y_2 + 120y_3 = 0,
\end{aligned}
\tag{17}
$$

In Simulink (17) is first modelled as shown in Fig. 12.

For (18), S_1=1200 and S_2 =24 for t=[0, 0.2]. Numerical simulation results presented in Table 7 were obtained after experimenting with different solvers. Also, alternative model to Fig. 12 is presented in Fig. 13.

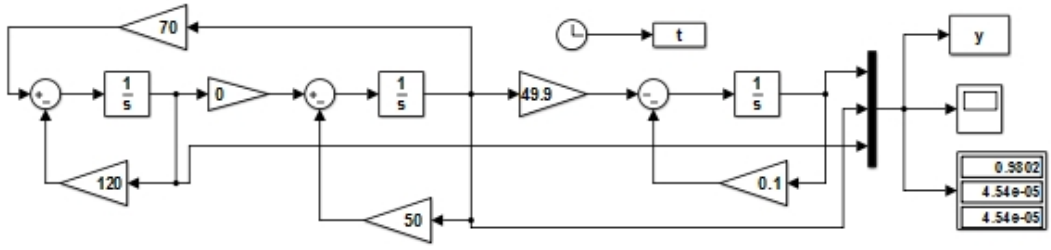

Fig. 12. Model I, Simulink model for example V

In MAPLE, the following code gave the analytical result:

```
restart;
PDEtools[declear]((x,y,z,f,g)(t),prime=t);
sys:=[diff(x(t),t)=-0.1*x(t)-49.9*y(t),diff(y(t),t)=-50*y(t),
```

diff(z(t),t)=70∗y(t)-120∗z(t), x(0)=2, y(0)=1, z(0)=2];
sol:=dsolve(sys);
odetest(sol,sys);

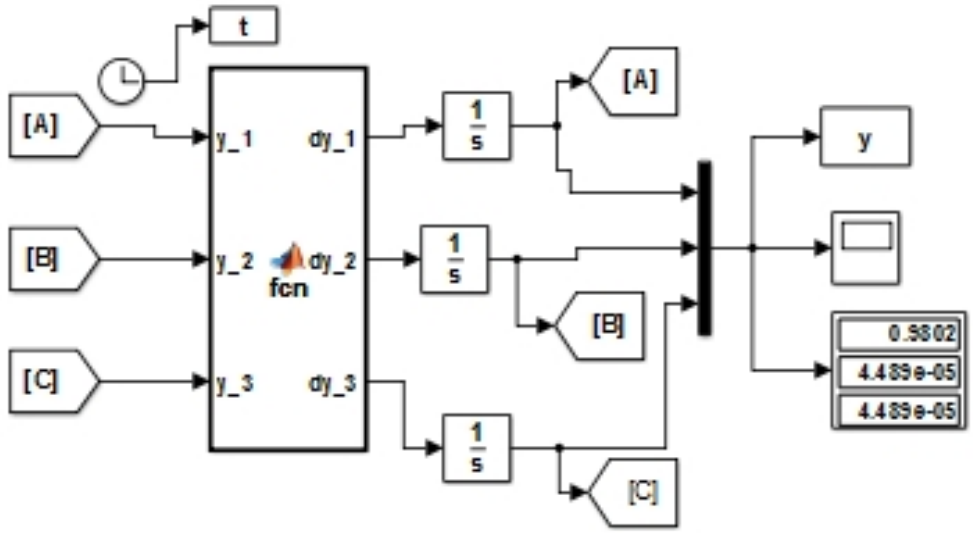

Fig. 13. Model II, Simulink model for example V

Thus, its closed-form solution is as given in (18) while the graphical solution is depicted in Fig. 14.

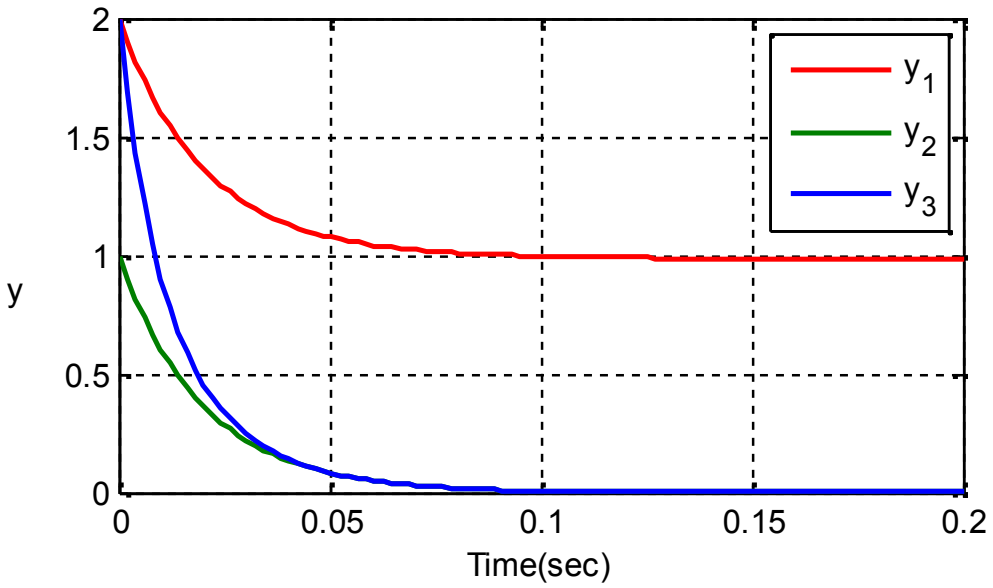

Fig. 14. Graphical Simulation result for example V

$$
\begin{aligned}
y_1 &= e^{-50t} + e^{-0.1t}, \\
y_2 &= e^{-50t}, \\
y_3 &= e^{-50t} + e^{-120t}.
\end{aligned} \tag{18}
$$

Example VI. The system given by (19) is considered [10].

$$
\begin{aligned}
y_1' &= -500000.5y_1 + 499999.5y_2, \\
y_2' &= 499999.5y_1 - 500000.5y_2,
\end{aligned} \quad y_1(0) = 0,\ y_2(0) = 2 \tag{19}
$$

In Simulink [11] and [12], (19) is modelled as given in Figs. 15 and 16.

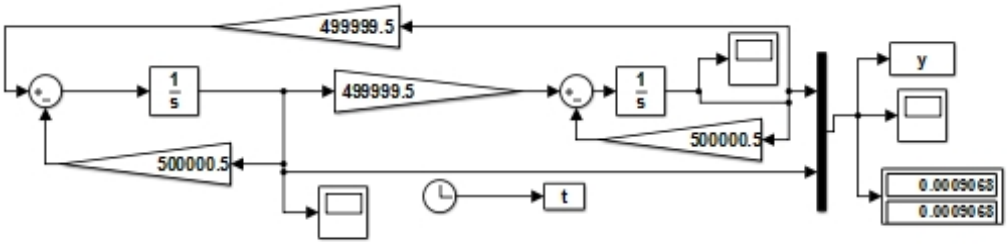

Fig. 15. Model I, Simulink model for Example VI

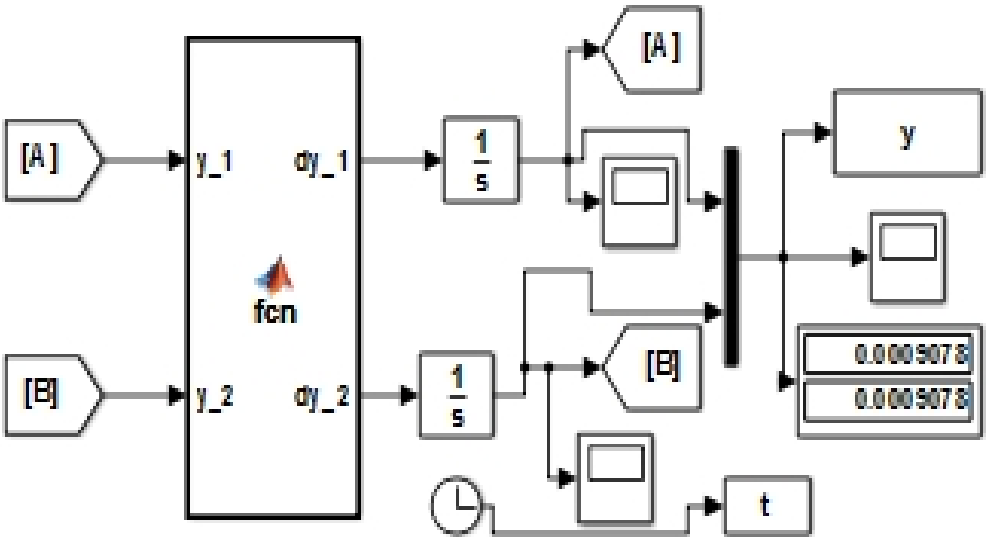

Fig. 16. Model II, Simulink model for Example VI

Equipt with our definitions for stiffness, S_1=10^6 and for t=[0, 7],S_2=7.0×10^6, hence numerical simulation gave the results in Table 8.

In MAPLE [13] the closed-form solution is obtained with the following:

```
restart;
PDEtools[declear]((x,y,z,f,g)(t),prime=t);
sys:=[diff(x(t),t)=-500000.5*x(t)-499999.5*y(t),
diff(y(t),t)=-499999.5*x(t)-500000.5*y(t),x(0)=0,y(0)=2];
sol:=dsolve(sys);
odetest(sol,sys);
```

The above gave the closed form solution in (20) and the graphical result is shown in Fig. 17.

$$y_1 = e^{-t} - e^{1000000t},$$
$$y_2 = e^{-t} + e^{-1000000t}. \tag{20}$$

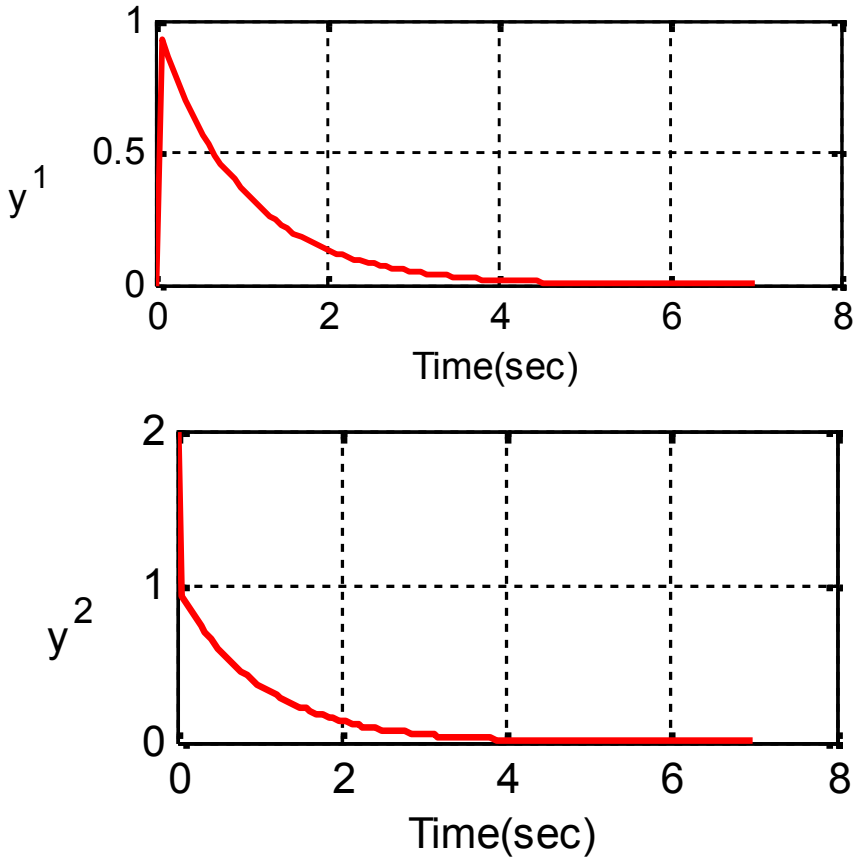

Fig. 17. Simulation result for Example VI

5. DISCUSSION OF RESULTS

For all the examples considered, it is worth noting that there exists no disparity, first between the graphical results of the two methods of simulation in Simulink. Secondly, graphical comparison between the numerical simulation in MATLAB and the symbolic simulation in MAPLE, agree.

From Example I, S_2=5000 suggests a stiff ODE, this was confirmed by numerical simulation result in Table 1. All the non-stiff solvers performed poorly with *ode113* having the largest number of steps-3793 (*ode45* gave 1819 steps). While the *stiff solvers* [14] with relative ease, converged to a solution with *ode23tb* having the smallest number of steps-75.

Table 1. Computation steps for example I

Solvers	ode45	ode23	ode113	ode15s	ode23s	ode23t	ode23tb
Steps - Model I	1819	2403	3793	89	88	88	75
Steps - Model II	1819	2403	3785	89	88	88	75

In Example II, S_1=20 and S_2=60, both suggest non-stiff. Simulation result is Table 2 concurs, with *ode45* having the minimum effort for solution to converge with just 57 steps. While the stiff solvers performed poorly with *ode23tb* having the largest number of steps-99.

Table 2. Computational steps for example II

Solvers	ode45	ode23	ode113	ode15s	ode23s	ode23t	ode23tb
Steps-Model I	57	72	91	90	77	99	84
Steps- Model II	57	72	91	90	77	99	84

For Example III, S_1=101 and S_2=302. Both suggest non-stiff, meaning a non-stiff solver should just be the best to approximate the solution of the ODE. On the contrary, simulation results in Table 3 shows that the system will best be suited with a *stiff* solver for simulation, with *ode23tb* having 66 steps and *ode113* having 194 steps for solution to converge (*ode45* converged with 97 steps).

Table 3. Computational Steps for Example III

Solvers	ode45	ode23	ode113	ode15s	ode23s	ode23t	ode23tb
Steps-Model I	97	128	194	74	64	76	66
Steps-Model II	97	128	197	74	64	76	66

With Example IV, we decided to do a kind of fragmented analysis based on the states of the system and their various integration times for solution to converge to a steady state solution. For y_3 and y_4, S_2=8.4 which suggest non-stiff and the result in Table 4 agrees with it, with *ode45* having the minimum number of steps-51. Considering y_1, S_2=1440 which suggests stiffness and is confirmed with the result in Table 5; with *ode113* having 783 step, *ode45*with 368 steps and *ode23s*with 67 steps. Finally for this Example to be analysed is the state variable y_2, S_2=840, this suggests non-stiff but Table 6 contradicts it, here we have all the non-stiff solvers performing badly compared to their *stiff* counterparts. Hence it will be best simulated with *ode23s* which gave 66steps while the worst is *ode113* with 461 (*ode45* gave 218 steps).

Table 4. Considering y_3 and y_4, S_2=8.4

Solvers	*ode45*	*ode23*	*ode113*	*ode15s*	*ode23s*	*ode23t*	*ode23tb*
Steps-Model I	51	52	55	54	52	54	52
Steps- Model II	51	52	55	54	52	54	52

Table 5. Considering y_1, S_2=1440

Solvers	ode45	ode23	ode113	ode15s	ode23s	ode23t	ode23tb
Steps-Model I	368	486	783	81	67	89	75
Steps- Model II	368	486	783	81	67	89	75

Table 6. Considering y_2 with S_2=840

Solvers	ode45	ode23	ode113	ode15s	ode23s	ode23t	ode23tb
Steps-Model I	218	287	461	79	66	81	69
Steps- Model II	218	287	461	79	66	81	69

With Example V, the two definitions for stiffness index are already at conflict with each other, with S_1=1200 and S_2=24. Simulation result in Table 7 gives us the accepted nature of the system, which is non-stiff, with *ode45* having 52 steps for solution to converge.

Table 7. Computational steps for example V

Solvers	ode45	ode23	ode113	ode15s	ode23s	ode23t	ode23tb
Steps - Model I	52	53	57	57	53	68	63
Steps - Model II	52	53	57	57	53	68	63

Finally Example VI, with $S_1=10^6$ and $S_2=7.0\times10^{6.}$ Both values suggest stiffness of the dynamic system. Simulation result in Table 8 agrees with it, with *ode113* having 4363823 steps (*ode45* has 2109151 steps) and *ode23s* having just 77 steps for solution to converge. Note, for Model II in Simulink for this particular example, all the non-stiff solvers gave a random-noise like pattern as the graphical result.

Table 8. Computational steps for example VI

Solvers	ode45	ode23	ode113	ode15s	ode23s	ode23t	ode23tb
Steps - Model I	2109151	2785813	4363823	98	77	93	80
Steps - Model II	2109151	2785813	4363823	98	77	93	80

6. CONCLUSION

The most pragmatic opinion about *stiffness* is that *stiff* ODEs are ODEs where certain implicit methods, in particular backward differentiation methods, perform much better than explicit ones. On that base, we conclude that *Stiffness* coefficient of 1000 does not translate to the fact that a linear ODE is *stiff* in all cases. As we have shown with Examples V. Neither does stiffness coefficient less than 1000 always suggest non-stiffness, as we have shown with Example III and Example IV (considering the state y_2).

It is imperative that a suspected *stiff* ODE in the neighbourhood of *stiffness* coefficient of 1000 be subjected to an experimental simulation with different solvers in the categories of both *stiff* and *non-stiff*, only after then can one conclude on the true nature of the ODE. Conclusions drawn here are based on the *solver* that requires the minimum number of steps (less work) to converge to a solution for the ODE.

From Example VI, a *stiffness* coefficient of the magnitude 10^6predicts *stiffness* in a linear ODE without ambiguity. Also, the non-stiff solvers gave no distinctive graphical trajectory of the state.

Numerical results of the most appropriate solver in each example were compared with its analytical ones graphically, no noticeable disparity was observed. Also, the second option of modelling ODEs using MATLAB Function block proves to be a viable alternative to the traditional block method.

Results of simulation in MATLAB/Simulink (Models I and II) gave the same result as that of MAPLE without any graphical disparity for all examples.

FUTURE WORK

More examples need to be explored to further support the position in this research and simulation time for each *solver* to converge to a solution needs to be included in the tabular results.

ACKNOWLEDGMENT

The authors will like to specially appreciate Prof. S.O Mohammed, *Director-General*, National Space Research and Development Agency (NASRDA), for his invaluable support of R&D in CSTP. Also in appreciation is the F.A Agboola, *Director, ESS* of NASRDA.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Godfrey C. Onwubolu. Mechatronics: Princioles and Applications. Elsevier Butterworth-Heinemann, Great Britian; 2005. ISBN 0 7506 6379 0.
2. Shampine LF, Thompson S. Stiff Systems. Scholarpedia. 2007;2(3):2855.
3. Aliyu B. Kisabo, Osheku CA, Adetor MA Lanre, Aliyu Funmilayo A. Ordinary Differential Equations: MATLAB/Simulink® Solutions. International Journal of Scientific & Engineering Research. 2012;3:8. ISSN 2229-5518.
4. Byrne GD, Thompson S. Error Control Matters. Departments of Applied Mathematics and Chemical & Environmental Engineering Illinois Institute of Technology Chicago, IL 60616, USA; 2007. gbyrne@wi.rr.com.
5. Shampine LF, Corless RM. Initial value problems for ODEs in problem solving environments. Preprint submitted to Elsevier Preprint; 2000.
6. Steven C Chapra. Applied Numerical Methods with MATLAB for Engineers and Scientist, Third Edition. New York; 2012. ISBN 978-0-07-340110-2.

7. Lambert JD. Computational methods in ordinary differential equation. John Wiley & Sons; 1972.
8. Cash JR. On the integration of stiff systems of ODEs using modified extended backward differentiation formula. Computer Math Applic. 1980;9(5):645-657.
9. Cash JR. A FORTRAN subroutine for solution of first order system for ordinary differential equations. ACM Trans. 1992;18(2)156.
10. Wu XY, Xiu JL. Two low accuracy methods for stiff systems. Appl Math Comput. 2001;123:141-153.
11. The Math Works, Inc. SIMULINK, Model-Based and System-Based Design, Version 5 @ COPYRIGHT 1990 – 2002. www.mathworks.com
12. The Math Works, Inc. Simulink, Getting Started Guide R2012b, version 8, © COPYRIGHT 1990–2012. www.mathworks.com
13. Frank Garvan. The MAPLE Book, Chapman & Hall/CRC. Boca Raton, London, New York, Washington, DC; 2002. ISBN 1-58488-232-8.
14. Holly Moore. MATLAB for Engineers, Third Edition. Pearson Education, Inc., publishing as Prentice Hall, One Lake Street, Upper Saddle River, New Jersey 07458; 2012. ISBN-13: 978-0-13-210325-1

Peer-review history:
The peer review history for this paper can be accessed here:
http://www.sciencedomain.org/review-history.php?iid=493&id=22&aid=4321