BULETINUL INSTITUTULUI POLITEHNIC DIN IAŞI
Tomul LI (LV), Fasc. 1–2, 2005
CONSTRUCŢII. ARHITECTURĂ

SOME TECHNIQUES FOR SOLVING "STIFF" EQUATIONS

BY
VICTOR-OCTAVIAN ROŞCA

The Structural Dynamics involves a large amount of computational effort. Most dynamic structural models require the solution of a set of 2^{nd} order differential equations. There are developed integration techniques for the 1^{st} order and 2^{nd} order differential equation. The 2^{nd} order set of equationis is submitted to a transformation [12] in order to obtain the first order system.

This paper deals with the "stiff" systems of 1^{st} order differential equations. From the physical point of view the stiff system consists of two components – one with a fast dynamic behavior and the other one, slow. The ignoring of the high frequency component may lead to wrong results.

There are presented some advanced solution methods, criteria for choosing the appropriate techniques and a case study.

1. Introduction

From the physical point of view the "stiff" system consists of two components – one with a fast dynamic behavior and the other one, slow. By ignoring the high frequency component it is possible to obtain wrong results.

Let us consider the following system of equations [2], where x_1 and x_2 are time (t) variables:

$$x_1' = 998x_1 + 1,998x_2, \quad x_2' = -999x_1 - 1,999x_2. \tag{1}$$

and the initial conditions:

$$x_1(0) = 1, \quad x_2(0) = 0. \tag{2}$$

By changing the variables

$$x_1 = 2y - z, \quad x_2 = -y + z. \tag{3}$$

it results the solution

$$x_1 = 2e^{-t} - e^{-1,000t}, \quad x_2 = -e^{-t} + e^{-1,000t}. \tag{4}$$

In [12] there are extensively presented many methods for the solution (*i.e.* single step techniques, like Euler, 2nd and 4th order Runge-Kutta, Fehlberg, the Richardson extrapolation and the Bulirsch-Stoer method; also multistep methods: Adams-Bashford, Rosenbrock and Shampine) of the system (1).

If one uses un-avoided the previously mentioned methods to integrate the system (1), the term that contains the $e^{-1,000t}$ needs an integration time step with a length of $h \ll 1/1,000$ to obtain the stability. This phenomenon occurs even the value of $e^{-1,000t}$ can be completely neglected for the computation of x_1 and x_2 immediately near the 0 point, as is depicted in the Fig. 1. This is the disadvantage of the "stiff" systems of equations – we are obliged to follow the solution path by taking into account the smallest step in order to achieve the stability – although the final solution can be obtained by an algorithm with a bigger time step.

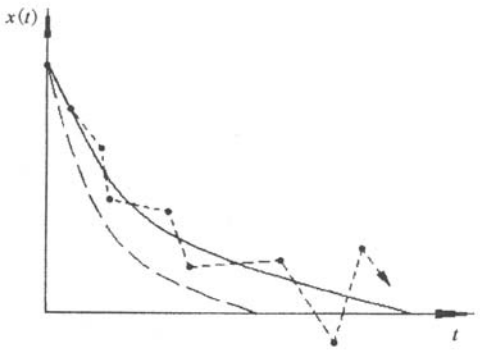

Fig. 1.– The instability phenomenon that occurs during the integration of "stiff" equations.

In the Fig. 1 it is depicted the solution of such a differential equation, consisting of two components, one marked with solid line and the other with a dotted line. The solution stability depends on the dotted line, that converges faster to 0.

2. The Solution Techniques

In the Fig. 2 it is presented the Simulink model of the problem (1) with the solution provided by (4), by means of the Matlab programming environment. Although on the entire domain the solution behaves well, in the vicinity of the start point some numerical problems occur, as it can be noticed in the zoom window of the 0...0.1 s range.

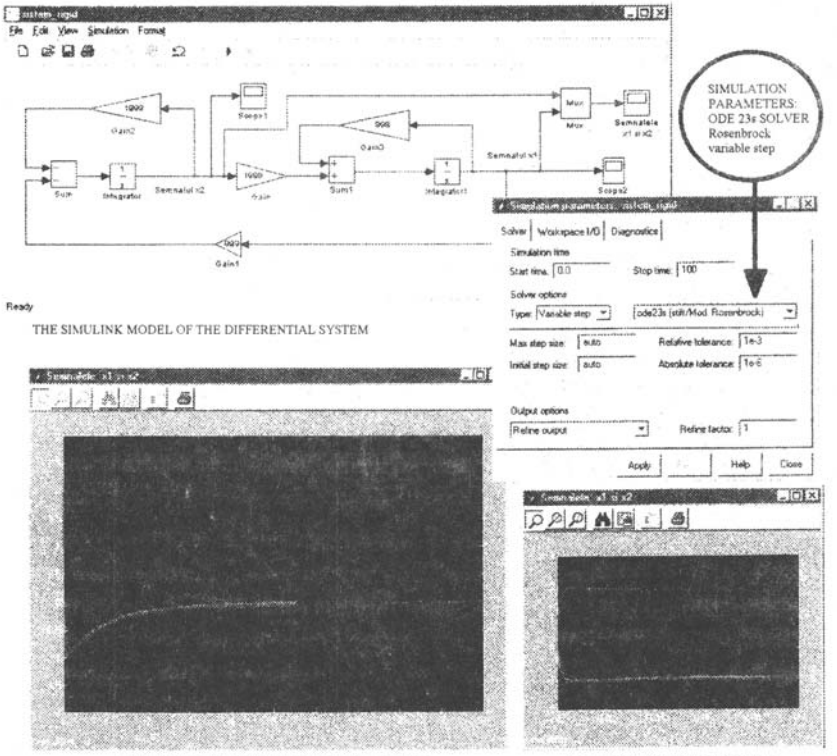

Fig. 2.– The Simulink model.

In order to find out the solution there is considered the case of a single equation

$$x' = -px, \tag{5}$$

where $p > 0$ is a constant coefficient. The explicit Euler integration scheme for (5) – with constant time step h – is:

$$x_{n+1} = x_n + hx_n' = (1 - ph)x_n. \tag{6}$$

The method becomes unstable if $h > 2/p$, because in this case $|x_n| \to \infty$ as $n \to \infty$.

The simplest method for the solution in this case is the implicit integration, in which the right-hand side is evaluated at the new location, x. We apply the backwards Euler scheme,

$$x_{n+1} = x_n + hx_{n+1}', \tag{7}$$

or:

$$x_{n+1} = \frac{x_n}{1 + ph}. \tag{8}$$

The method is unconditionally stable even $h \to \infty$, $x_{n+1} \to \infty$ and this is the correct solution of the differential equation; if we analyse the physical/mechanical phenomenon, the implicit method converges to the real solution – the equilibrium. This important feature of the implicit methods is applicable only to the linear system, but also, in the general case, the implicit methods offer a better stability. This is the case of the implicit methods with the 1^{st} order of precision. To solve the "stiff" systems of equations one may use some methods provided with higher orders of precision. There are three important classes of higher order integration techniques for the stiff system:

a) Generalizations of the Runge-Kutta method, from which the best is the Rosenbrock method. The first implementations were carried out by K a p s and R e n t r o p [17].

b) Generalizations of the B u l i r s c h - S t o e r [13] method, particularly the semi-implicit B a d e r and D e u f l h a r d [19] extrapolation method.

c) The predictor–corrector methods, most of them derived from the G e a r [9] differentiation method.

There will be discussed the implementations of the first two classes. The systems of equations are depending explicitly on time, t, on the right-hand side $f(X,t)$; they are transformed by adding t to the variable vector

$$\begin{pmatrix} X \\ t \end{pmatrix}' = \begin{pmatrix} f \\ 1 \end{pmatrix}. \tag{9}$$

It is very important to scale correctly the variables from the "stiff" systems. Like the case of the common systems, it has to be taken a vector, X_{scal}, in order to scale the errors. For instance, to obtain constant errors in a fraction format, X_{scal} is scaled by $|X|$. In rigid problems often appear components of the solution that decay fast so can be eliminated. It is advisable to control the relative error from P; a possible way is to apply a limit value such as

$$X_{\text{scal}} = \max(P, |X|). \tag{10}$$

The Rosenbrock methods are easily implemented. They are recommended for the systems that need a medium precision for the solution ($\varepsilon \leftarrow$ E-4 ... E-5 in the error criterion) and for small sizes (ten equations). In case of large systems the semi-implicit extrapolation methods are recommended.

The Rosenbrock method proposes a solution of the form:

$$X(t_0 + h) = t_0 + \sum_{i=1}^{s} c_i k_i, \tag{11}$$

where the correction factors, k_i, are found by solving the s linear equations system:

$$(1 - \gamma h f')k_i = hf\left(X_0 + \sum_{j=1}^{i-1} \alpha_{ij} k_j\right) + hf' \sum_{j=1}^{i-1} \gamma_{ij} k_j, \quad (i = 1, ...s). \tag{12}$$

By f' was denominated above the Jacobi matrix. The γ, c_i, α_{ij} and γ_{ij} coefficients are independently constants having fixed values for a given problem. If $\gamma = \gamma_{ij} = 0$, then the method becomes a classical Runge-Kutta one. The equations (12) can be successively solved for $k_1, k_2, ...$

The main element that leads to good result is the algorithm for the automatic adjustment of the step size. Kaps and Rentrop (*op. cit.*) proposed an optimized Runge-Kutta-Fehlberg method that lies on the computation of two estimators for (11), a "real" X and one of a smaller order, $\widehat{X}$, that have the coefficients $\widehat{c}_i$, $(i=1, ..., \widehat{s})$, with different values, where $\widehat{s} < s$ and the k_i coefficients are the same. The difference between X and $\widehat{X}$ leads to the estimation of a round off error, that can be used as control parameter for the length of the time step. Kaps and Rentrop showed that the smallest value for s that improves the solution is $s=4$ and $\widehat{s}=3$. These values lead to a IVth order numerical method. In order to reduce to minimum the number of matrix products between the matrices and vectors of the right side of (12), the equations are re-writen under the form

$$g_i = \sum_{j=1}^{i-1} \gamma_{ij} k_j + \gamma k_i. \tag{13}$$

The equations are now:

$$\begin{cases} \left(\dfrac{1}{\gamma h} - f'\right) g_1 = f(X_0), \\ \left(\dfrac{1}{\gamma h} - f'\right) g_2 = f(X_0 + a_{21} g_1) + \dfrac{c_{21} g_1}{h}, \\ \left(\dfrac{1}{\gamma h} - f'\right) g_3 = f(X_0 + a_{31} g_1 + a_{32} g_2) + \dfrac{c_{31} g_1 + c_{32} g_2}{h}, \\ \left(\dfrac{1}{\gamma h} - f'\right) g_4 = f(X_0 + a_{41} g_1 + a_{42} g_2 + a_{43} g_3) + \dfrac{c_{41} g_1 + c_{42} g_2 + c_{43} g_3}{h}. \end{cases} \tag{14}$$

Kaps and Rentrop offered two sets of parameters which provide different stability characteristics. There are provided also other sets, which a very used one is that of S h a m p i n e [18] and is implemented in the Matlab/Simulink Software.

The semi-implicit extrapolation method is applied instead of the Bulirsch-Stoer method (that partitions the differential equation by the modified midpoint rule) that doesn't work for "stiff" problems. Bader and Deuflhard (*op. cit.*) elaborated a semi-implicit partition that works very well and leads to an extrapolation just like in the case of the Bulirsch-Stoer method.

The start point is represented by an implicit formula of the midpoint rule:

$$X_{n+1} - X_{n-1} = 2hf\left(\frac{X_{n+1} + X_{n-1}}{2}\right). \tag{15}$$

Eq. (15) is converted in the semi-implicit form by the linearization of the right side around of $f(X_n)$. The result is the semi-implicit rule of the midpoint

$$\left(1 - h\frac{\partial f}{\partial X}\right) X_{n+1} = \left(1 + h\frac{\partial f}{\partial X}\right) X_{n-1} + 2h\left[f(X_n) - \frac{\partial f}{\partial X} X_n\right]. \tag{16}$$

In this method a special value is used for the first step size, which is the value from the semi-implicit Euler method and a step with a re-calculated step size, where the last X_n is replaced by:

$$\overline{X}_n = \tfrac{1}{2}(X_{n+1} + X_{n-1}). \tag{17}$$

Bader and Deuflhard demonstrated that the error series involve only even powers of h. For the simplification of the programming process it is recommended the re-writing of the above equations using the notation $\Delta_k \equiv X_{k+1} - X_k$. Considering $h = H/m$, the integration process starts with:

$$\Delta_0 = \left(1 - h\frac{\partial f}{\partial X}\right)^{-1} hf(X_0), \quad X_1 = X_0 + \Delta_0. \tag{18}$$

Then, for $k = 1, ..., m-1$, one calculates:

$$\Delta_k = \Delta_{k-1} + 2\left(1 - h\frac{\partial f}{\partial X}\right)^{-1} [hf(X_k) - \Delta_{k-1}], \quad X_{k+1} = X_k + \Delta_k. \tag{19}$$

Finally the expression

$$\Delta_m = \left(1 - h\frac{\partial f}{\partial X}\right)^{-1} [hf(X_m) - \Delta_{m-1}], \quad \overline{X}_m = X_m + \Delta_m \tag{20}$$

is obtained.

The sequence for the computation of the time step lengths is

$$n = 2, 6, 10, 14, 22, 34, 50, ..., \tag{21}$$

where each member differs from its predecessor by the smallest multiple of 4 that makes the ratio of the successive terms be $\leftarrow 5/7$.

3. Case-study

We consider a simple case of a SDOF dynamic system in free damped motion, with the following law of motion:

$$\ddot{x} + 100\dot{x} + 0.9999x = 0, \tag{22}$$

and initial conditions $x(0) = 1$, $\dot{x}(0) = 0$. For the system with the governing equation (22) the Simulink model is shown in Fig. 3. The above equation is transformed from the t-space into s-space by the means of the Laplace transform, where by $X(s)$ was denoted the Laplace transform of $x(t)$, with $X(s) = Lx(t)$

$$s^2X(s) - sx(0) - \dot{x}(0) + 100sX(s) - 100x(0) + 0.9999X(s) = 0. \tag{23}$$

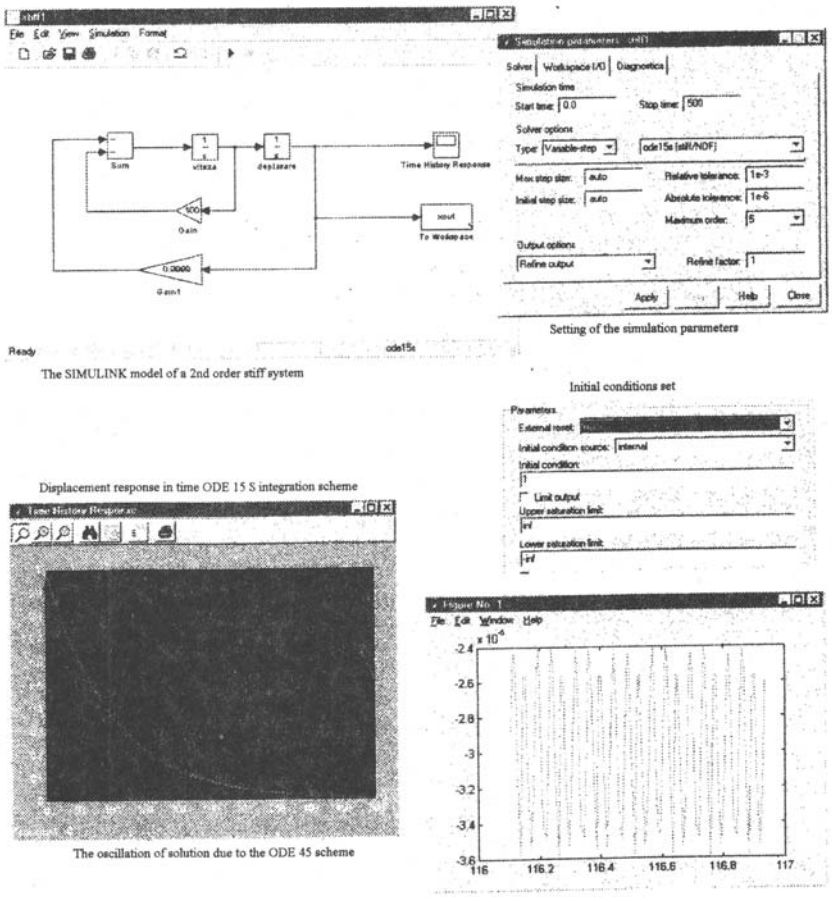

Fig. 3.– The oscillation of the "stiff" system solution.

The transfer function of the system becomes:

$$H(s) = \frac{1}{s^2 + 100s + 0.9999} \tag{24}$$

and the output in the s-space is

$$X(s) = (s + 100)H(s). \tag{25}$$

Applying the inverse Laplace transform one obtains the solution:

$$x(t) = +1.0001\mathrm{e}^{-0.01t} - 0.0001\mathrm{e}^{-99.99t}. \tag{26}$$

The system response has two components. The first component has a magnitude of four orders greater than the second one but also decays 10,000 times slower. The second member has a much smaller magnitude (0.0001) and decays fast. Thus, the slower response has the predominant role.

Let us focus on the Simulink model from Fig. 3, where are depicted the model and the initial conditions. The Integrator block has the initial condition set on 0 and the integrator block for displacement has the initial condition value of 1 set on the Dialog box. In the Simulation Parameters block the start time is set to 0 and stop time is set to 500. It was used an "x_{out}" transfer block of the output parameters to the Matlab workspace. The simulation has as goal the analysis of several integration algorithms for the 1^{st} order differential equations, implemented for th solution of these "stiff" systems.

4. Results. Conclusions

Initially, the simulation was performed with the ODE 15S routine. The result is very quickly obtained, in few hundreds of seconds (CPU time). The output vectors to Matlab contain approximately 150 elements, computed at the automatically fixed time steps. If the model is run by using an ODE 23S integrator, designed on the basis of the Rosenbrock method, the runtime is reduced to 0.002 s. The results are presented in the Table 1.

Table 1

The Runtimes of Several Matlab Routines Involved in the Model Analysis

Routine (solver)	Time step size	Runtime CPU time, [s]
ODE 23 S (Rosenbrock method)	variable, automatically adaptive	0.002
15 S	variable, automatically adaptive	0.06
23 S (stiff systems - trapezoidal)	variable, automatically adaptive	0.06
ODE 45 (Dormand - Prince)	variable, automatically adaptive	3.30
ODE 113 (Adams-Bashford)	variable, automatically adaptive	4.29
ODE 4 (Runge-Kutta of IV^{th} order)	4 s, fixed	Error - Divergent Analysis
ODE 4 (idem)	0.01 s, fixed	5.38

The Simulink software uses the ODE 45 routine as the implicit integrator (based on the Dormand - Prince algorithm). After the computation with this routine one may notice the same output window (in the complete time domain 0, ..., 500 s) like in te case of ODE 23S. Unlike in the other cases, the run is performed in 3.30 s and the automatically adjusted time step produces the output vectors with 15,000 elements.

A zooming window of 1 s length was selected, depicted in the bottom-right corner of the Fig. 3. In comparison to the bottom-left frame, that shows the global response, one may notice a high frequency component (with a fast dynamic) in the response trajectory, of a 10^{-5} order magnitude. This proves the fact that the performances of such a kind of analysis are highly dependent on an adequate solution algorithm.

Finally, a mention regarding the inappropriate choice of an ODE 4 solver with a fixed time step (which is the 4^{th} order Runge-Kutta algorithm with fixed step) leads to the collapse. For this model, the setting of a 4 s time step, which is equivalent to 125 time steps for analysis (equal to the number of the used time steps by the Rosenbrock ODE 23 S algorithm), produces the divergence. The simulation is stopped and an error message is produced (caused by the singularity). The numerical experiments evidenced that the Runge-Kutta routine needs a time step of 0.01 s to attain the convergence.

Received, June 10, 2005

"Gh.Asachi" Technical University, Jassy,
Department of Structural Mechanics

REFERENCES

1. Collatz L., *The Numerical Treatment of Differential Equations.* Springer-Verlag, NY, 1966.
2. Crandall S. H., *Engineering Analysis.* Mc-Graw-Hill Book Co., NY, 1956.
3. Fröberg C. E., *Introduction to Numerical Analysis.* Addison-Westley Publ. Co. Inc., Reading, Mass., 1969.
4. Kuo B.C., *Automatic Control Systems.* Englewood Cliffs, Prentice Hall, NJ, 1995.
5. Dabney J., *The Student Edition of Simulink - Dynamic System Simulation for Matlab.* Prentice-Hall, NJ, 1998.
6. Dodescu Gh., Toma M., *Metode de calcul numeric.* Edit. Did. şi Pedag., Bucureşti, 1976.
7. Nistor I., *Analiză Numerică.* Univ. Tehn. „Gh. Asachi", Iaşi, 1997.
8. Abramowitz M., Stegun I. A., *Handbook of Mathematical Functions.* Appl. Math. Series, **55**, National Bureau of Standards, Washington DC, 1964.
9. Gear W., *Numerical Initial Value Problems in Ordinary Differential Equations.* Englewood Cliffs, NJ, 1971.
10. Shampine L., Watts H. A., *Mathematical Software* III. Appl. Math. and Comp., **5**, 257-275 (1979).
11. Rice J. R., *Numerical Methods. Software and Analysis.* McGraw-Hill, NY, 1983.
12. Roşca V.-O., *Optimizarea programelor de calcul al structurilor supuse acţiunilor dinamice şi seismice.* Ph. D. Diss., "Gh. Asachi" Techn. Univ., Jassy, 2002.
13. Stoer J., Bulirsch R., *Introduction to Numerical Analysis*, Springer-Verlag, NY, 1980.
14. Forsythe G.E., Malcolm M., Moler C., *Computer Methods for Mathematical Computations.* Englewood Cliffs, Prentice-Hall, NJ, 1977.
15. Cash J.R., Karp A., *A Variable Order Runge-Kutta Method for Initial Value Problems with Rapidly Varying Right-hand Sides.* ACM Trans. on Mathem. Software, **16**, 201-222 (1990).

16. D e u f l h a r d P., *Recent Progress in Extrapolation Methods for Ordinary Differential Equations.* SIAM Review, **27**, *4*, 505-535 (1985).
17. K a p s P., R e n t r o p P., *Generalized Runge-Kutta Methods of Order Four with Stepsize Control for Stiff Ordinary Differential Equations.* Numerische Mathematik, **38**, 55-68 (1979).
18. S h a m p i n e L.F., *Implementation of Rosenbrock Methods.* ACM Trans. on Mathem. Software, **8**, 93-113.
19. B a d e r G., D e u f l h a r d P., *A Semi-implicit Mid-Point Rule for Stiff Systems of ODE.* Numerische Mathematik, **41**, 373-398 (1983).
20. S h a m p i n e L.F., R e i c h e l t M.W., *The MATLAB ODE Suite.* The Mathworks, Inc., Natick, Massachusetts, 1996, www.mathworks.com.

METODE PENTRU REZOLVAREA ECUAŢIILOR „RIGIDE"

(Rezumat)

Dinamica structurilor presupune un volum important de calcul numeric. Cele mai multe modele dinamice conduc la rezolvarea unui sistem de ecuaţii diferenţiale de gradul al doilea. Se utilizează pe scară largă metodele pentru rezolvarea sistemelor de ecuaţii diferenţiale de gradul I şi II. Sistemele de gradul al doilea se transformă prin [12] într-un sistem de gradul I.

Se prezintă câteva metode pentru rezolvarea sistemelor aşa-zise „rigide" (de ordinul I) de ecuaţii diferenţiale. Din punct de vedere fizic un sistem rigid este alcătuit din două componente – una cu o comportare dinamică rapidă iar cealaltă cu o comportare lentă. Dacă se neglijează componenta cu frecvenţă ridicată atunci pot apare erori numerice de calcul.

Sunt prezentate câteva metode avansate de rezolvare, criterii pentru alegerea unei metode adecvate precum şi un studiu de caz.