Computer Assisted Mechanics and Engineering Sciences, 6: 479–486, 1999.

Mixed algorithm for solving boundary value problem

Béla Paláncz and György Popper
Technical University of Budapest, Budapest 1521, Hungary

(Received March 8, 1999)

Symbolic computation has been applied to Runge–Kutta technique in order to solve two-point boundary value problem. The unknown initial values are considered as symbolic variables, therefore they will appear in a system of algebraic equations, after the integration of the ordinary differential equations. Then this algebraic equation system can be solved for the unknown initial values and substituted into the solution. Consequently, only one integration pass is enough to solve the problem instead of using iteration technique like shooting-method. This procedure is illustrated by solving the boundary value problem of the mechanical analysis of a liquid storage tank. Computation was carried out by MAPLE V. Power Edition package.

1. Introduction

There are many engineering models represented by ordinary differential equations with split boundary value problems.

Shooting and finite difference methods [1–2], trial function expansions based on variational principles or weighted-residual methods as well as different type of collocation and quasi-linearization [3] and perturbation [4] techniques, have been widely used for a long time to solve such problems.

Computer algebra systems like MACSYMA, REDUCE, MAPLE, MATHEMATICA and in certain extend other type of systems as MATLAB and MATHCAD, give possibility to carry out not only numerical but also symbolical computations.

Many traditional algorithms can be improved, sometimes considerably, via imbedding symbolic parts into the numerical algorithm. These hybrid techniques involving numerical as well as symbolic manipulations, provide arbitrary precision defeating instability problems and reduce the number of iterations in general.

The application of hybrid techniques to boundary value problems was studied in [5] for the case of second and third order, linear and non-linear ordinary differential equations including eigenvalue and stiffness problems. Some partial differential equations were also tacled. To solve the illustrative examples, the MUMATH programming system was employed.

In this paper the methods proposed by [5], the so called slope retention techniques have been extended for non-autonom, linear system of differential equations using symbolic Runge–Kutta method.

2. Boundary value problem

Let us consider a linear, non-autonom differential equation system of n variables, in matrix form:

$$\frac{\mathrm{d}\,\mathrm{y}(x)}{\mathrm{d}x} = \mathrm{A}(x)\,\mathrm{y}(x) + \mathrm{b}(x)$$

where A is a matrix of n^2 dimensions, $y(x)$ and $b(x)$ are vectors of n dimensions, and x is a scalar independent variable.

In case of boundary value problem, the value of some dependent variables, y_j, are not known at the beginning of the integration interval, at $x = x_1$, but they are given at the end of this interval, at $x = x_2$.

The usually employed methods need subsequent integration of the system, because their trial-error technique or they require to solve large linear equation system, in case of discretization methods. In this paper a new technique is presented, which is based on the symbolical evaluation of the well known Runge–Kutta method. This technique needs only one integration of the differential equation system and a solution of the linear equation system representing the boundary conditions at $x = x_2$.

3. Symbolic Runge–Kutta method

The well known fourth-order Runge–Kutta method, in our case, can be represented by the following formulas:

$$R1_i = \mathrm{A}(x_i)\,\mathrm{y}(x_i) + \mathrm{b}(x_i)$$

$$R2_i = \mathrm{A}\left(x_i + \frac{h}{2}\right)\left(\mathrm{y}(x_i) + \frac{R1_i\,h}{2}\right) + \mathrm{b}\left(x_i + \frac{h}{2}\right)$$

$$R3_i = \mathrm{A}\left(x_i + \frac{h}{2}\right)\left(\mathrm{y}(x_i) + \frac{R2_i\,h}{2}\right) + \mathrm{b}\left(x_i + \frac{h}{2}\right)$$

$$R4_i = \mathrm{A}(x_i + h)\,(\mathrm{y}(x_i) + R3_i\,h) + \mathrm{b}(x_i + h)$$

and then the value of $y_{i+1} = \mathrm{y}(x_i + h)$ can be computed as:

$$y_{i+1} = \mathrm{y}(x_i) + \frac{(R1_i + 2\,(R2_i + R3_i) + R4_i)\,h}{6}\,.$$

A symbolic system, like MAPLE, is able to carry out this algorithm not only with numbers but symbols. It means that the unknown elements of $\mathrm{y}(x1)$ can be considered unknown symbols. These symbols will appear in every y_i, as well as in $\mathrm{y}(x2)$, too.

The following MAPLE procedure can carry out this symbolic computation:

```
> restart;
> RKSymbolic:=proc(x0,y0,A,b,M,N,h)
> local R1,R2,R3,R4,y,i,j,ylist;
> with(linalg);
> ylist:=vector([0$M]);
> y:=evalm(y0);
> for j from 1 to M do
> ylist[j]:=[[x0,y0[j]]] od;
> for i from 1 to N do
> R1:=evalm(A(x0+i*h)&*y+b(x0+i*h));
> R2:=evalm(A(x0+i*h+h/2)&*(y+R1*h/2)+b(x0+i*h+h/2));
> R3:=evalm(A(x0+i*h+h/2)&*(y+R2*h/2)+b(x0+i*h+h/2));
> R4:=evalm(A(x0+i*h+h)&*(y+R3*h)+b(x0+i*h+h));
> y:=evalm(y+(R1+2*(R2+R3)+R4)*h/6);
> for j from 1 to M do
> ylist[j]:=[op(ylist[j]),[x0+i*h,y[j]]] od;
> od;
> [evalm(y),ylist]:
> end:
```

Let us consider a simple illustrative example. The differential equation is:

$$\frac{\mathrm{d}^2\,\mathrm{y}(x)}{\mathrm{d}x^2} - \left(1 - \frac{x}{5}\right)\mathrm{y}(x) = x\,.$$

The prespecified boundary values are:

$$\mathrm{y}(1) = 2 \qquad \text{and} \qquad \mathrm{y}(3) = -1.$$

Introducing

$$\mathrm{y1}(x) = \mathrm{y}(x) \qquad \text{and} \qquad \mathrm{y2}(x) = \frac{\mathrm{d}\,\mathrm{y}(x)}{\mathrm{d}x}\,.$$

The matrix form of the differential equation is:

$$\begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}x}\,\mathrm{y1}(x) \\ \frac{\mathrm{d}}{\mathrm{d}x}\,\mathrm{y2}(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 - \frac{1}{5}\,x & 0 \end{bmatrix} \begin{bmatrix} \mathrm{y1}(x) \\ \mathrm{y2}(x) \end{bmatrix} + \begin{bmatrix} 0 \\ x \end{bmatrix}$$

```
> A:=x->matrix([[0,1],[(1-x/5),0]]);
```

$$A := x \to \begin{bmatrix} 0 & 1 \\ 1 - \frac{1}{5}\,x & 0 \end{bmatrix}$$

```
> b:=x->vector([0,x]);
```

$$b := x \to [0,\ x]$$

```
> y0:=vector([2,s]);
```

$$y0 := [2,\ s]$$

```
> x0:=1;
```

$$x0 := 1$$

The unknown initial condition for $y2$ is s. The order of the system $M = 2$. Let us consider the number of the integration steps, $N = 10$, so the step size is $h = 1/5$.

```
> M:=2;N:=10;h:=2./N;
```

$$M := 2$$

$$N := 10$$

$$h := .2000000000$$

```
> ysol:=RKSymbolic(x0,y0,A,b,M,N,h):
```

The function values at the different x_i are, for $y1_i$:

```
> sol1:=ysol[2][1];
```

$$\begin{aligned} sol1 := [[1,\ 2],\ & [1.200000000,\ 2.055334186 + .2009866667\,s], \\ & [1.400000000,\ 2.226107320 + .4077215240\,s], \\ & [1.600000000,\ 2.521649911 + .6255151090\,s], \\ & [1.800000000,\ 2.953940376 + .8592959466\,s], \\ & [2.000000000,\ 3.537291287 + 1.113676121\,s], \\ & [2.200000000,\ 4.288011455 + 1.392978103\,s], \\ & [2.400000000,\ 5.224018115 + 1.701225944\,s], \\ & [2.600000000,\ 6.364375819 + 2.042103061\,s], \\ & [2.800000000,\ 7.728741088 + 2.418878236\,s], \\ & [3.000000000,\ 9.336694830 + 2.834301450\,s]] \end{aligned}$$

The list contains the $(x_i,\ y1_i)$ pairs, and every $y1_i$ values depends on the unknown initial conditions.

Consequently, we have got symbolic results using the traditional numerical Runge–Kutta method.

4. Mixed algorithm for solving boundary value problem

In order to compute the proper values of the unknown initial values, the boundary conditions at $x = x2$ can be applied. In our case $y1(3) = -1$. We can get $y1(3)$ from the solution, therefore this condition can be written as:

```
> eq:=ysol[1][1]=-1;
```

$$eq := 9.336694830 + 2.834301450\,s = -1$$

Let us solve this equation numerically:

```
> sol:=fsolve(eq,s);
```

$$sol := \{s = -3.646999097\}$$

```
> assign(sol);
```

Now this s value can be substituted into the solutions for $y1$ and $y2$:

```
> ySol:=map(eval,ysol[2]):
```

Then we get the numerical solution, for example, for $y1$:

```
> ySol[1];
```

```
[[1, 2], [1.200000000, 1.322335994], [1.400000000, .739147290],
    [1.600000000, .240396873], [1.800000000, -.179911165],
    [2.000000000, -.524284521], [2.200000000, -.792178429],
    [2.400000000, -.980351367], [2.600000000, -1.083172200],
    [2.800000000, -1.092905654], [3.000000000, -1.000000000]]
```

The truncation error can be decreased by using smaller step-size, h, and the round-off error can be controlled by the *Digits* command of MAPLE.

5. Analysis of a liquid storage tank

Let us consider a cylindrical liquid storage tank, where the thickness of wall/radius ratio is small enough to ensure membrane stress state, see Fig. 1.

The four differential equations describing the deflection, $w(x)$, rotation, $\alpha(x)$, bending moment, $M_b(x)$, and transverse shear force, $F_Q(x)$ distributions along the length of the storage are the followings, [6]:

$$\frac{\mathrm{d}\,\mathrm{w}(x)}{\mathrm{d}x} = \alpha(x)\,, \qquad \frac{\mathrm{d}\,M_b(x)}{\mathrm{d}x} = F_Q(x)\,,$$

$$\frac{\mathrm{d}\,\alpha(x)}{\mathrm{d}x} = -\frac{M_b(x)}{No\,\delta(x)^3}\,, \qquad \frac{\mathrm{d}\,F_Q(x)}{\mathrm{d}x} = Do\,\delta(x)\,\mathrm{w}(x) - \gamma\,x\,,$$

where

$$No = \frac{E\,{t_o}^3}{12\,(1-\nu^2)}\,, \qquad Do = \frac{E\,t_o}{R^2}\,, \qquad \text{and} \qquad \delta(x) = \frac{\mathrm{t}(x)}{t_o}\,.$$

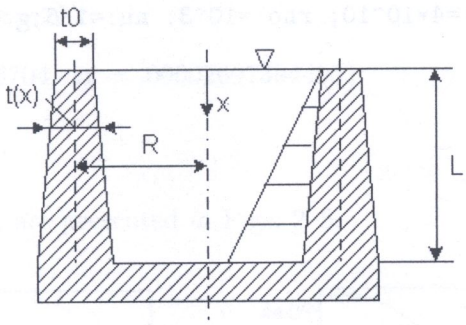

Fig. 1. Liquid storage tank

The boundary conditions are:

$$\mathrm{w}(L) = 0\,, \qquad \alpha(L) = 0\,, \qquad M_b(0) = 0\,, \qquad \text{and} \qquad F_Q(0) = 0\,.$$

Let us suppose that the thickness of the wall is linear function of x, namely

$$\mathrm{t}(x) = \frac{1 + \frac{x}{L}}{2}\,.$$

Introducing variables y_i, $i = 1, 2, 3, 4$:

$$y_1(x) = \mathrm{w}(x)\,, \qquad y_3(x) = M_b(x)\,,$$
$$y_2(x) = \alpha(x)\,, \qquad y_4(x) = F_Q(x)\,,$$

one can get

$$\frac{\mathrm{d}\, y_1(x)}{\mathrm{d}x} = y_2(x)\,,$$
$$\frac{\mathrm{d}\, y_2(x)}{\mathrm{d}x} = -\frac{8\, y_3(x)}{No\,(1 + \frac{x}{L})^3}\,,$$
$$\frac{\mathrm{d}\, y_3(x)}{\mathrm{d}x} = y_4(x)\,,$$
$$\frac{\mathrm{d}\, y_4(x)}{\mathrm{d}x} = \frac{Do\,(1 + \frac{x}{L})\, y_1(x)}{2} - \gamma\, x\,,$$

and

$$y_1(L) = 0, \qquad y_2(L) = 0, \qquad y_3(0) = 0, \qquad y_4(0) = 0.$$

The matrix form of the system is:

$$\begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}x}\, y_1(x) \\ \frac{\mathrm{d}}{\mathrm{d}x}\, y_2(x) \\ \frac{\mathrm{d}}{\mathrm{d}x}\, y_3(x) \\ \frac{\mathrm{d}}{\mathrm{d}x}\, y_4(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -8\,\frac{1}{No\,(1+\frac{x}{L})^3} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{1}{2}\, Do\,(1 + \frac{x}{L}) & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g\,\rho\,x \end{bmatrix}$$

Let us consider the following data:

```
> R:=5;L:=1.5; t0:=0.1;E:=4*10^10; rho:=10^3; nu:=1/6;g:=9.81;
```

$$R := 5$$

$$L := 1.5$$

$$t0 := .1$$

$$E := 40000000000$$

$$\rho := 1000$$

$$\nu := \frac{1}{6}$$

$$g := 9.81$$

Now,

```
> No:=E*t0^3/12/(1-nu^2);
```

$$No := .3428571428\,10^{7}$$

```
> Do:=E*t0/R^2;
```

$$Do := .1600000000\,10^{9}$$

Then:

$$A := x \rightarrow \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -8\,\frac{1}{No\,(1+\frac{x}{L})^3} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{1}{2}\,Do\,(1+\frac{x}{L}) & 0 & 0 & 0 \end{bmatrix}$$

```
> b:=x->vector([0,0,0,-g*rho*x]);
```

$$b := x \rightarrow [0,\ 0,\ 0,\ -g\,\rho\,x]$$

```
> y0:=vector([s1,s2,0,0]);
```

$$y0 := [s1,\ s2,\ 0,\ 0]$$

```
> x0:=0;
```

$$x0 := 0$$

where $s1$ and $s2$ are the unknown initial conditions.

```
> M:=4;N:=100;
```

$$M := 4, \qquad N := 100$$

```
> ysol:=RKSymbolic(x0,y0,A,b,M,N,L/N):
> eq1:=ysol[1][1];
```

$$eq1 := -8.417839115\,s1 - 1.594827552\,s2 + .0002993784801$$

```
> eq2:=ysol[1][2];
```

$$eq2 := -6.518553710\,s2 - 13.39539692\,s1 + .0006569052057$$

```
> sol:=fsolve({eq1,eq2},{s1,s2});
```

$$sol := \{s2 = .00004534418704,\ s1 = .00002697394405\}$$

```
> assign(sol);
> y:=map(eval,ysol[2]):
```

Then, the solutions for the y_i are presented in Figs. 2–5.

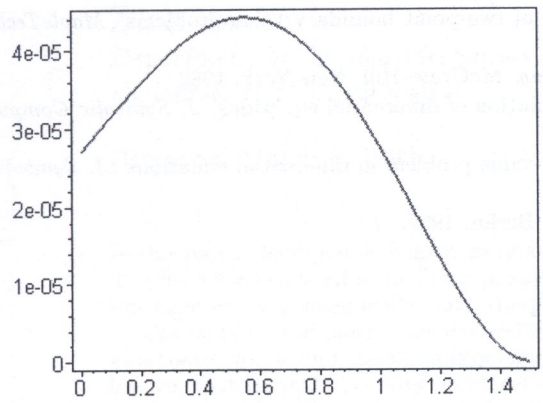

Fig. 2. Deflection vs. storage height

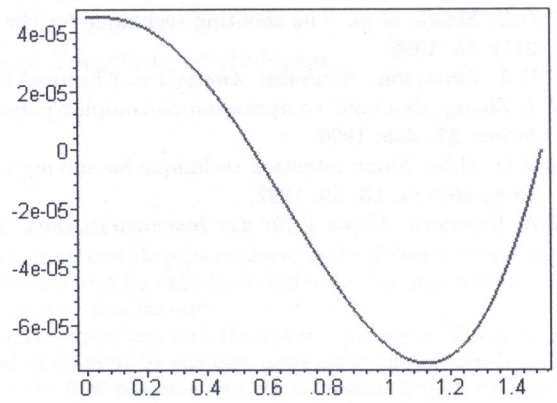

Fig. 3. Rotation vs. storage height

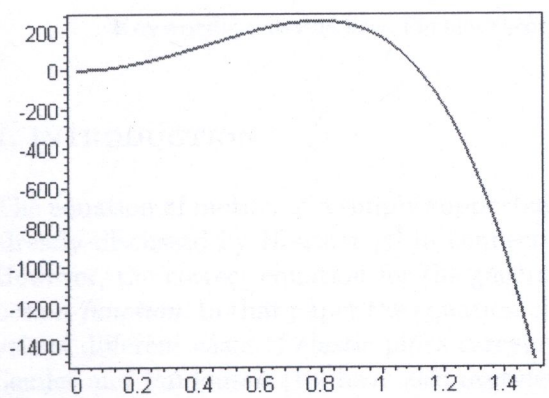

Fig. 4. Bending moments vs. storage height

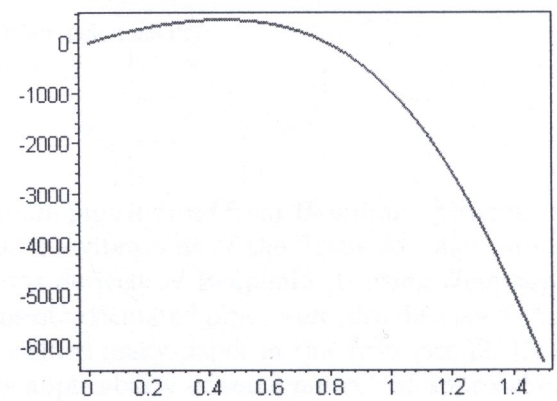

Fig. 5. Shear force vs. storage height

6. Concluding remarks

The extended form of slope retention technique can be applied to solving of linear boundary value problems of non-autonom linear differential equation systems. To do that, symbolic Runge–Kutta technique can be employed followed by numerical solution of a linear algebra system representing the boundary conditions at the end of the integration interval.

Although the method needs only one integration pass, because of symbolic operation, the computation speed will slow down comparing with the pure numerical integration. However, in the future the speed of the symbolic calculation could be increased considerably through software or direct hardware developments similarly to the graphical operations.

A mechanical example was given to demonstrate this method, which seems to be useful especially in case of systems with many unknown initial conditions.

Acknowledgements

This work was supported by the Hungarian National Scientific and Research Foundation (OTKA T 029771 and T 025258).

References

[1] B. Birkeland. *Mathematics with Mathcad.* Chartwell-Bratt Ltd., Studentlitteratur, Sweden, 1997.

[2] D.B. Meade *et al.* The shooting technique for the solution of two-point boundary value problems. *MapleTech*, **3**(1): 85, 1996.

[3] B.A. Finlayson. *Nonlinear Analysis in Chemical Engineering.* McGraw–Hill, New York, 1980.

[4] J. Zhang. Symbolic computation on complex polynomial solution of differential equations. *J. Symbolic Computation*, **22**: 345, 1996.

[5] R.D. Mills. Slope retention technique for solving boundary-value problem in differential equations. *J. Symbolic Computation*, **13**: 59, 1992.

[6] A. Krawietz. *Maple V für das Ingenieurstudium.* Springer, Berlin, 1997.