Bull. Nov. Comp. Center, Num. Anal., 12 (2003), 73–80

The program SIMODE for solution of ODE systems with singular matrix multiplying the derivative*

A.I. Levykin

The paper presents an algorithm for the numerical solution of the initial value problems for systems of ordinary differential equations with singular matrix multiplying the derivative. The algorithm uses the (m, k) scheme of the Rosenbrock type with time-lagging derivative matrices, and the adaptive step size control for the global error. Some examples of solution of test problems are presented.

The routine `SIMODE` finds an approximation to the solution of a system of differential equations $A(y)y' = f(y)$, on $[0, x_{\text{end}}]$, y given, with a solution-dependent, singular square matrix A. Non-autonomous problems may also be included formally adding the equation for the independent variable, $x' = 1$. We assume that $A(y)$ and $f(y)$ have sufficiently many bounded derivatives, and that the initial value is *consistent*:

- $f(y(0))$ is in the range of $A(y(0))$;
- $A(y)$ has constant rank in neighbourhood of the solution $y(x)$;
- the matrix pencil $\{A + \lambda B\}$, with $B = \frac{\partial(f(y)-A(y)y')}{\partial y}$ has index of nilpotency 1 along the solution.

Thus, the conditions guarantee the existence of a unique, smooth solution $y(x)$ [1].

The routine uses the (m, k) formulas [2, 3]

$$x_{n+1} = x_n + \sum_{i=1}^{m} \mu_i k_{xi}, \quad y_{n+1} = y_n + \sum_{i=1}^{m} \mu_i k_{yi},$$

where the internal stages are carried out by

$$\begin{aligned} D_n k_{xi} &= \eta A_2(k_{x(i-1)} + \sum_{j\in J_i} \alpha_{ij} k_{xj}) + (\eta - 1)hA_1(k_{x(i-1)} + \sum_{j\in J_i} \gamma_{ij} k_{xj}), \\ k_{yi} &= \frac{1}{ah}(k_{xi} - \eta(k_{x(i-1)} + \sum_{j\in J_i} \alpha_{ij} k_{xj}) - (1 - \eta)h(k_{y(i-1)} + \sum_{j\in J_i} \gamma_{ij} k_{yj}). \end{aligned}$$

*Supported by the Russian Foundation for Basic Research under Grant 01-07-90367.

Here a, μ_i, β_{ij}, α_{ij}, and γ_{ij} are parameters defining the stability and the accuracy properties, h is the integration step, A_1, A_2 are matrices approximating the derivatives

$$F_{ny} = \frac{\partial F(x_n, y_n)}{\partial y}, \quad F_{nx} = \frac{\partial F(x_n, y_n)}{\partial x}, \quad D_n = A_2 + ahA_1,$$

where $F(x_n, y_n) = A(x_n, y_n)z_n - f(x_n, y_n)$, z_n is the approximation of y' at the point x_n.

The algorithm involves embedded pair of schemes of second and third orders using the time-lagging derivative matrices and attempts to keep the global error proportional to a user-specified tolerance. This routine is efficient for stiff systems with index 1 or index 0.

At every step once decomposition of a matrix D_n is evaluated, the function of a right side of a differential problem is 3 times calculated, backward in the Gauss method is 5 times executed.

SIMODE (Double precision)

Purpose Solves a first order differential-algebraic system of equations, $A(y)y' = f(y)$, using the Rosenbrock-type methods.

Usage `CALL SIMODE(MS, N, T, TK, H, HM, EP, TR, Y, YPR, WK, IWK, GCN, DGCN)`

Arguments

`MS` – the integer work array of length 11.

`MS(1)` – an indicator to the first call; 0 means the first call for the problem (initialization will be done); 1 means that the first call is performed (Input/Output).

`MS(2)` – an indicator specifying the task to be performed; 0 means "take one step only and return"; 1 means the normal computation of output values of `Y(T)` at `T` = `TK` (Input).

`MS(3)` – an indicator responsible for the method calculating the matrix of partial derivatives of $g(t, y, y')$. At `MS(3)` = 0, the matrix is numerically calculated using the DGCN, at `MS(3)` = 1 (Input).

`MS(4)` – not used.

`MS(5)` – this indicator is used to signal a singular or poorly conditioned partial derivative matrix encountered during the factor phase. If the value is nonzero, the routine returns control to the user. Default value is 0 (Output).

MS(6) – the number of steps taken for the problem so far (Input/Output).

MS(7) – the number of g evaluations for the problem so far (Input/Output).

MS(8) – the number of the derivative matrix evaluations for the problem so far (Input/Output).

MS(9) – the number of the matrix LU decompositions for the problem so far (Input/Output).

MS(10) – the number of inverse motions in the Gauss method (Input/Output).

MS(11) – the number of repeated calculations of the solution for the problem so far (Input/Output).

N – the number of differential equations (Input).

T – an independent variable. In input, T is used only for the first call, as the initial point of integration. In output, after each call, T is the value at which a computed solution Y is evaluated if MS(2) $= 0$. Or T $=$ TK if MS(2) $= 1$ (Input/Output).

TK – the end point of integration (Input).

H – the step size to be attempted at the first step. The default value is determined by the solver. In output, H takes on the value of the step predicted (Input/Output).

HM – the minimum absolute step size allowed. If the step predicted is less than HM, H $=$ HM, the computational accuracy is not controlled. The default value is determined by the solver (HM $= 10^{-12}$) (Input).

EP – a relative error tolerance parameter (Input).

Y – an array of dependent variables. In the first call, Y should contain initial values. In output, after each call, Y contains the computed solution evaluated at T if MS(2) $= 0$. Or Y(T) $=$ Y(TK) if MS(2) $= 1$ (Input/Output).

YPR – an array of size N containing the derivative values y'. In the first call, YPR should contain initial values $y'(t_0)$ so that $g(t_0, y, y') = 0$. In output, after each call, YPR contains a derivative of the computed solution evaluated at T if MS(2) $= 0$, or YPR(T) $=$ YPR(TK) if MS(2) $= 1$ (Input/Output).

WK – the real work array of length $15\text{N} + 3\text{N}^2$.

IWK – the integer work array of length N.

TR – a parameter. If $|$Y(i)$| >$ TR, the relative error EP will be controlled in Y(i). If $|$Y(i)$| <$ TR, the absolute error EP*TR will be controlled in Y(i). The default value is determined by the solver (TR $= 1.0$) (Input).

GCN – the user-supplied subroutine to evaluate the function $g(t, y, y')$. The usage is CALL GCN(N, T, Y, YPR, GVAL), where GCN has the form

```
subroutine gcn(n, t, y, ypr, gval)
double precision t, y, ypr, gval
dimension y(n), ypr(n), gval(n)
..................
gval(i) = g(i)
..................
return
end
```

Here N, T, Y, and YPR are the input parameters, and the array GVAL of size N containing the function values $g(t, y, y')$ is the output. Y, YPR, and GVAL are arrays of length N. GCN must be declared EXTERNAL in the calling program.

DGCN – the name of a user-supplied subroutine to compute partial derivatives of $g(t, y, y')$. It is to have the form:

```
subroutine dgcn(n, t, y, ypr, pdy, pdypr, pdt)
double precision t, y, ypr, pdy, pdypr, pdt
dimension y(n), ypr(n), pdy(n, n), pdyrp(n, n), pdt(n)
..................
dpy(i, j) = dg(i) / dy(j)
..................
dpypr(i, j) = dg(i) / dy'(j)
..................
pdt(i) = dg(i) / dt
..................
return
end
```

Here N, T, Y, and YPR are input, and the arrays PDY, PDYPR, PDT are to be loaded with nonzero partial derivatives in the output: PDY is $\partial g/\partial y$, PDYPR is $\partial g/\partial y'$, and PDT is $\partial g/\partial t$. DGCN should be declared EXTERNAL in the calling program.

Example 1. The following is a simple test problem (the Van der Pol equation, see Example 1 for the routine DASPG from the IMSF library), with the coding needed for its solution by SIMODE. The test problem is solved as differential-algebraic system and has $n = 2$ equations:

$$g_1 = y_2 - y_1' = 0,$$
$$g_2 = (1 - y_1^2)y_2 - \epsilon(y_1 + y_2') = 0$$

on the interval from $t = 0$ to $t_k = 26$, with the initial conditions $y_1 = 2$, $y_2 = -2/3$, $y_1' = y_2$, $y_2' = 0$ for the value $\epsilon = 0.2$.

```
      double precision t, tk, h, hm, ep, tr, y, ypr, wk
      external dgcn, gcn
      dimension y(2), ypr(2), wk(42), iwk(2), ms(11)
      data ms/0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0/
  910 format(8x,'t',10x,'y1',10x,'y2',10x,'ypr1',8x,'ypr2')
  920 format(/,1x,'Number of RP  calls with SIMODE = ', i10)
  930 format(1x,'Number of DRP calls with SIMODE = ', i10)
  940 format(2x,5d12.4)
      n = 2
      tr = 1.d-0
      ep = 1.d-3
      hm = 1.d-12
      t = 0.d0
      tk = 26.d0
      h = 1.d-5
      y(1) = 2.d0
      y(2) = -2.d0 / 3.d0
      ypr(1) = y(2)
      ypr(2) = 0.d0
      write(*,910)
      write(*,940) t, y, ypr
    1 call simode(ms, n, t, tk, h, hm, ep, tr, y, ypr, wk,
     &  iwk ,gcn, dgcn)
      if(t .lt. tk) goto 1
      write(*, 940) t, y, ypr
      write(*, 920) ms(7)
      write(*, 930) ms(8)
      stop
      end

      subroutine gcn(n, t, y, ypr, g)
      double precision g, ypr, t, y, eps
      dimension y(2), ypr(2), g(2)
      data eps /.2d0/
      g(1) = y(2) - ypr(1)
      g(2) = (1.0d0 - y(1)**2)*y(2) - eps*(y(1) + ypr(2))
      return
      end
```

```
      subroutine dgcn(n, t, y, ypr, dgy, dgypr, dgt)
      double precision t, y, ypr, dgy, dgypr, dgt, eps
      dimension y(1), ypr(1), dgy(n, 1), dgypr(n, 1), dgt(1)
      data eps /.2d0/
      dgy(1, 2) = 1.d0
      dgy(2, 1) = -eps - 2.d0 * y(1) * y(2)
      dgy(2, 2) = 1.d0 - y(1) ** 2
      dgypr(1, 1) = -1.d0
      dgypr(2, 2) = -eps
      return
      end
```

Output

```
      t            y1          y2          ypr1         ypr2
   .0000D+00   .2000D+01  -.6667D+00  -.6667D+00   .0000D+00
   .2600D+02   .1483D+01  -.2341D+00  -.2343D+00  -.8274D-01

 Number of RP  calls with SIMODE =         416
 Number of DRP calls with SIMODE =          82
```

Example 2. The SIMODE is used to solve the so-called pendelum problem (see Example 2 for the routine DASPG from the IMSF library). The problem has $n = 5$ equations:

$$\begin{aligned} g_1 &= y_3 - y_1' = 0, \\ g_2 &= y_4 - y_2' = 0, \\ g_3 &= -y_1 y_5 - m y_3' = 0, \\ g_4 &= -y_2 y_5 - mg - m y_4' = 0, \\ g_5 &= m(y_3^2 + y_4^2) - mg y_2 - l^2 y_5 = 0 \end{aligned} \tag{1}$$

and is solved on the interval from $t = 0$ to π, with the initial conditions $y_1 = l$, $y_i = 0$, $i = 2, \ldots, 5$, $y_i' = 0$, $i = 1, \ldots, 5$. All parameters of the pendelum are the same as for the corresponding parameters for Example 2 for the routine DASPG. In this example, we use the option MS(3)=1 for the numerical computation of partial derivatives.

```
      double precision t, tk, h, hm, ep, tr, y, ypr, wk,
     & maxten, tmax
      external dgcn, gcn
      dimension y(5), ypr(5), wk(150), iwk(5), ms(11)
      data ms/0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0/
      data pi/3.14159265359d0/
  910 format(1x,5e12.5)
```

```
920 format(1x,'Number of steps are fulfilled by SIMODE = ',
   &i10)
930 format(1x, 'Extreme string tension of', d10.3,
   &'(lb/s**2)', 2x,'occurred at time', d10.3, /)
    n = 5
    tr = 1.d-0
    ep = 1.d-3
    tk = pi
    hm = 1.d-12
    h = 1.d-5
    t = 0.d0
    y(1) = 2.d0
    maxten = 0.d0
  1 call simode(ms, n, t, tk, h, hm, ep, tr, y, ypr, wk,
   &  iwk, gcn, dgcn)
    if(dabs(maxten) .lt. dabs(y(5))) then
    maxten =  y(5)
    tmax = t
    end if
    if(t .lt. tk) goto 1
    maxten = maxten * 2.20462d0
    write(*, 930) maxten, tmax
    write(*, 920) ms(6)
    stop
    end

    subroutine gcn(n, t, y, ypr, g)
    double precision g, ypr, t, y, meterl, masskg, lensq,
   &  mg, grav
    dimension y(n), ypr(5), g(5)
    logical first
    data  first /.true./
    data  grav /9.80665d0/
    if (first) go to 20

10  g(1) = y(3) - ypr(1)
    g(2) = y(4) - ypr(2)
    g(3) = -y(1)*y(5) - masskg*ypr(3)
    g(4) = -y(2)*y(5) - masskg*ypr(4) -mg
    g(5) = masskg * (y(3)**2 + y(4)**2) - mg*y(2) -
   &       lensq*y(5)
    return
```

```
20  masskg = 98.d0 * .4536d0
    meterl = 6.5d0 * .3048d0
    lensq = meterl ** 2
    mg = masskg * grav
    first = .false.
    go to 10
    end

    subroutine dgcn(n, t, y, ypr, pdy, pdypr, pdt )
    double precision t, y, ypr, dx, dy, dt
    dimension y(1), ypr(1), pdy(n, 1), pdypr(n, 1), pdt(1)
    return
    end
```

Output

```
Extreme string tension of  .153D+04 (lb/s**2)
occurred at time  .251D+01
The number of steps are fulfilled by SIMODE = 47
```

References

[1] Haier E., Wanner G. Solving ordinary differentials equations II. – Springer-Verlag, 1996.

[2] Levykin A.I., Novikov E.A. One-step method of the three order accuracy for solving implicit systems of the ordinary differential equations // Modelirovanie v Mekhanike. – 1989. – Vol. 3, № 4. – P. 90–101 (in Russian).

[3] Levykin A.I., Novikov E.A. A class of (m, k)-methods for solving implicit systems // Soviet Math. Dokl. – 1996. – Vol. 348, № 4. – P. 442–445 (in Russian).