Bull. Nov. Comp. Center, Num. Anal., 11 (2002), 95–101

The program NODE for solution of ODE stiff systems

E.A. Novikov

The paper introduces an algorithm for the numerical solution of initial value problems for systems of ordinary differential equations (ODE). The algorithm uses the Rosenbrock-type and the Runge–Kutta-type schemes with Jacobian freezing and automatic step size control policy based on the global error estimation. Some examples of solution of test problems are given.

The routine NODE solves systems of first order ordinary differential equations of the form $y' = f(y)$ with initial data given. The integration algorithm involves:

- 10 Rosenbrock-type methods [1–5]

$$y_{n+1} = y_n + \sum_{i=1}^{m} p_i k_i, \qquad D_n = E - ahf'_n, \qquad f'_n = \frac{\partial f(y_n)}{\partial y},$$

$$D_n k_i = hf\Big(y_n + \sum_{j=1}^{i-1} \beta_{ij} k_j\Big) + \sum_{j=1}^{i-1} \alpha_{ij} k_j.$$

  For $f(y) = Ay + b$, the schemes are of the order from two to ten, for a nonlinear function $f(y)$ – from two to three. The methods are used with the freezing of the Jacobi matrix to be calculated both analytically and numerically.

- 13 explicit Runge–Kutta-type methods [6–9]

$$y_{n+1} = y_n + \sum_{i=1}^{m} p_i k_i, \qquad k_i = hf\Big(y_n + \sum_{j=1}^{i-1} \beta_{ij} k_j\Big).$$

  The domains of the stability methods are extended to 200 along the real axis. With explicit methods, the scheme order and the number of stages are automatically chosen depending on the step size and the value of eigenvalues of the Jacobi matrix.

With the explicit Runge–Kutta-type methods and the Rosenbrock-type formulae, the choice of a scheme is subject to the estimate of a maximum eigenvalue of the Jacobi matrix [8]. This estimate is defined by the power method with the use of the stages already calculated [6, 7].

NODE (Double precision)

Purpose Solves an initial-value problem for ordinary differential equations using the Rosenbrock and the Runge–Kutta methods.

Usage `CALL NODE(MS, N, T, TK, H, HM, EP, Y, F, M, TR, RP, DRP)`

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 Jacobi matrix. At `MS`(3) = 0 the matrix is numerically calculated using the DRP, at `MS`(3) = 1 (Input).

`MS(4)` – an indicator responsible for the integration method. At `MS`(4) = 0, the Rosenbrock-type methods and explicit methods with automatic choice of numerical scheme are used and at `MS`(4) = 1 the explicit Runge-Kutta methods are used (Input).

`MS(5)` – an indicator responsible for the problem type: `MS`(5) = 0 if the problem is nonlinear, `MS`(5) = 1 if the problem is linear (Input).

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

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

`MS(8)` – the number of Jacobian 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 the repeated calculations of the solution for the problem so far (Input/Output).

`N` – the size of the ODE system (the number of first order ordinary 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).

F – the real work array of length 2N(8 + N).

M – 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).

RP – the user-supplied subroutine to evaluate functions. It is to have the form:

```
subroutine rp(n, t, y, c)
double precision t, y, c
dimension y(n), c(n)
.................
c(i) = f(i)
..................
return
end
```

where N, T, and Y are the input parameters, and the array C=f(y) is the output. Y and C are arrays of length N. RP must be declared EXTERNAL in the calling program.

DRP – the name of the user-supplied subroutine to compute Jacobian matrix. It is to have the form:

```
subroutine drp(n, t, y, a)
double precision t, y, a
dimension y(n), a(n, n)
..................
a(i, j) = df(i) / dy(j)
..................
return
end
```

where N, T, and Y are input parameters and the array A is to be loaded with partial derivatives (elements of Jacobian matrix) on output. The DRP must be declared EXTERNAL in the calling program.

Example 1. Let us consider a simple example problem (Enright and Pryce, 1987) with the coding needed for its solution by the NODE. The test problem has $n = 2$ equations:

$$\begin{aligned} y_1' &= -y_1 - y_1 y_2 + 294 y_2, \\ y_2' &= -3y_2 + 0.01020408(1 - y_2)y_1 \end{aligned}$$

on the interval from $t = 0$ to 240, with the initial conditions $y_1 = 1$, $y_2 = 0$.

```
    double precision t, tk, h, hm, ep, tr, y, f
    external rp, drp
    dimension y(2), f(40), m(2), ms(11)
    data ms/0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0/
910 format(1x,4d15.7)
920 format(1x,'Number of RP  calls with NODE = ', i10)
930 format(1x,'Number of DRP calls with NODE = ', i10)
    n    = 2
    t    = 0.d0
    tk   = 240.d0
    h    = 1.d-2
    hm   = 1.d-12
    ep   = 1.d-3
    tr   = 1.d0
    y(1) = 1.d0
    y(2) = 0.d0
    write(*, 910) t, (y(j), j = 1, n)
 10 continue
    call node(ms, n, t, tk, h, hm, ep, y, f, m, tr, rp, drp)
    write(*, 910) t, (y(j), j = 1, n)
    if(dabs(t - tk) .gt. 1.d-12) goto 10
```

```
write(*, 920) ms(7)
write(*, 930) ms(8)
stop
end

subroutine rp(n, t, y, f)
double precision t, y, f
dimension y(1), f(1)
f(1) = - y(1) - y(1) * y(2) + 294.d0 * y(2)
f(2) = -3.d0 * y(2) + 0.01020408d0 * (1 - y(2)) * y(1)
return
end

subroutine drp(n, t, y, a)
double precision t, y, a
dimension y(1), a(n, 1)
a(1, 1) = -1.d0 - y(2)
a(1, 2) =  -y(1) + 294.d0
a(2, 1) = 0.01020408d0 * (1.d0 - y(2))
a(2, 2) = -3.d0 - 0.01020408d0 *  y(1)
return
end
```

Output

```
         t               y1              y2
   0.0000000D+00   0.1000000D+01   0.0000000D+00
   0.2400000D+03   0.3913376D+00   0.1330194D-02
 Number of RP  calls with NODE =       46
 Number of DRP calls with NODE =        8
```

Example 2. The NODE was used to solve the so-called Robertson problem

$$\begin{aligned} y_1' &= -0.04y_1 + 10^4 y_2 y_3, \\ y_2' &= -y_1' - y_3', \\ y_3' &= 3 \cdot 10^7 y_2^2 \end{aligned}$$

on the interval from $t = 0$ to 10, with the initial conditions $y_1 = 1$, $y_2 = 0$, $y_3 = 0$.

```
double precision t, tk, h, hm, ep, tr, y, f
external rp, drp
dimension y(3), f(66), m(3), ms(11)
data ms/0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0/
```

```
910 format(1x,5d15.7)
920 format(1x,'Number of RP  calls with NODE = ', i10)
930 format(1x,'Number of DRP calls with NODE = ', i10)
    n    = 3
    t    = 0.d0
    tk   = 10.d0
    h    = 1.d-5
    hm   = 1.d-12
    ep   = 1.d-2
    tr   = 1.d-5
    y(1) = 1.d0
    y(2) = 0.d0
    y(3) = 0.d0
    write(*, 910) t, (y(j), j = 1, n)
 10 continue
    call node(ms, n, t, tk, h, hm, ep, y, f, m, tr, rp, drp)
    write(*, 910) t, (y(j), j = 1, n)
    if(dabs(t - tk) .gt. 1.d-12) goto 10
    write(*, 920) ms(7)
    write(*, 930) ms(8)
    stop
    end

    subroutine rp(n, t, y, f)
    double precision t, y, f
    dimension y(1), f(1)
    f(1) = -.04d0 * y(1) + 1.d4 * y(2) * y(3)
    f(3) = 3.d7 * y(2) * y(2)
    f(2) = -f(1) - f(3)
    return
    end

    subroutine drp(n, t, y, a)
    double precision t, y, a
    dimension y(1), a(n, 1)
    return
    end
```

Output

```
      t              y1             y2             y3
 0.0000000D+00  0.1000000D+01  0.0000000D+00  0.0000000D+00
 0.1000000D+02  0.8406192D+00  0.1619187D-04  0.1593646D+00
Number of RP  calls with NODE =         91
Number of DRP calls with NODE =          0
```

References

[1] Novikov V.A., Novikov E.A., Umatova L.A. Frezing of the Jacobi matrix in the Rosenbrock type method of the second order accuracy // Proc. BAIL-IV Conf. – Bool Press, 1986. – P. 380–386.

[2] Novikov E.A., Shitov Yu.A., Shokin Yu.I. One-step iteration-free methods of solving stiff systems // Soviet Math. Dokl. – 1989. – Vol. 38, № 1. – P. 212–216.

[3] Novikov E.A. Construction of the (m, k)-methods for the solution of linear systems of ordinary differential equations // Mathematical models and tools for Chemical Kinetics. AMSE Transactions 'Scientific Siberian', Series A. – 1993. – Vol. 9. – P. 88–103.

[4] Novikov E.A., Golushko M.I., Shitov Yu.A. The freeze of the Jacobi matrix in the (m, k)-methods of order three // Advances in Modeling & Analysis, A. – AMSE Press, 1995. – Vol. 28, № 1. – P. 41–64.

[5] Novikov E.A., Golushko M.I., Shitov Yu.A. Approximation of Jacobi matrix in the (m, k)-methods of order three // Advances in Modeling & Analysis, A. – AMSE Press, 1995. – Vol. 28, № 3. – P. 19–40.

[6] Novikov V.A., Novikov E.A. On the accuracy and stability control of one-step methods of integration of ordinary differential equations // Proc. BAIL-III Conf. – Bool Press, 1984. – P. 81–93.

[7] Novikov V.A., Novikov E.A. Control of the stability of explicit one-step methods of integrating ordinary differential equations // Soviet Math. Dokl. – 1984. – Vol. 30, №. 1. – P. 211–215.

[8] Novikov E.A. Construction of algorithm for the integrating stiff differential equations on nonuniform schemes // Soviet Math. Dokl. – 1984. – Vol. 30, № 2. – P. 358–361.

[9] Novikov V.A., Novikov E.A. Explicit methods of Runge-Kutta type with adaptive stability region // Proc. BAIL-V Conf. – Bool Press, 1988. – P. 269–276.