Solve an initial-value problem for ordinary differential equations using either Adams-Moulton's or Gear's BDF method.

Usage

CALL IVPAG (IDO, N, FCN, FCNJ, A, T, TEND, TOL, PARAM, Y)

Arguments

IDO — Flag indicating the state of the computation. (Input/Output)

IDO	State
1	Initial entry
2	Normal re-entry
3	Final call to release workspace
4	Return because of interrupt 1
5	Return because of interrupt 2 with step accepted
6	Return because of interrupt 2 with step rejected
7	Return for new value of matrix A.

Normally, the initial call is made with IDO = 1. The routine then sets IDO = 2, and this value is then used for all but the last call that is made with IDO = 3. This final call is only used to release workspace, which was automatically allocated by the initial call with IDO = 1. See Comment 5 for a description of the interrupts.

When IDO = 7, the matrix *A* at *t* must be recomputed and IVPAG/DIVPAG called again. No other argument (including IDO) should be changed. This value of IDO is returned only if PARAM(19) = 2.

N — Number of differential equations. (Input)

FCN — User-supplied SUBROUTINE to evaluate functions. The usage is CALL FCN (N, T, Y, YPRIME), where
N — Number of equations. (Input)
T — Independent variable, *t*. (Input)
Y — Array of size N containing the dependent variable values, *y*. (Input)
YPRIME — Array of size N containing the values of the vector *y*′ evaluated at (*t*, *y*). (Output)
See Comment 3.
FCN must be declared EXTERNAL in the calling program.

FCNJ — User-supplied SUBROUTINE to compute the Jacobian. The usage is CALL FCNJ (N, T, Y, DYPDY) where

N — Number of equations. (Input)
T — Independent variable, *t*. (Input)
Y — Array of size N containing the dependent variable values, *y*(*t*). (Input)

DYPDY — An array, with data structure and type determined by PARAM(14) = MTYPE, containing the required partial derivatives ∂fi/∂yj. (Output)
These derivatives are to be evaluated at the current values of (*t*, *y*). When the Jacobian is dense, MTYPE = 0 or = 2, the leading dimension of DYPDY has the value N. When the Jacobian matrix is banded, MTYPE = 1, and the leading dimension of DYPDY has the value 2

* NLC + NUC + 1. If the matrix is banded positive definite symmetric, MTYPE = 3, and the leading dimension of DYPDY has the value NUC + 1.

FCNJ must be declared EXTERNAL in the calling program. If PARAM(19) = IATYPE is nonzero, then FCNJ should compute the Jacobian of the righthand side of the equation $Ay' = f(t, y)$. The subroutine FCNJ is used only if PARAM(13) = MITER = 1.

A — Matrix structure used when the system is implicit. (Input)
The matrix *A* is referenced only if PARAM(19) = IATYPE is nonzero. Its data structure is determined by PARAM(14) = MTYPE. The matrix A must be nonsingular and MITER must be 1 or 2. See Comment 3.

T — Independent variable, *t*. (Input/Output)
On input, T contains the initial independent variable value. On output, T is replaced by TEND unless error or other normal conditions arise. See IDO for details.

TEND — Value of *t* = *tend* where the solution is required. (Input)
The value *tend* may be less than the initial value of *t*.

TOL — Tolerance for error control. (Input)
An attempt is made to control the norm of the local error such that the global error is proportional to TOL.

PARAM — A *floating-point* array of size 50 containing optional parameters. (Input/Output)
If a parameter is zero, then the default value is used. These default values are given below. Parameters that concern values of the step size are applied in the direction of integration. The following parameters must be set by the user:

	PARAM	Meaning
1	HINIT	Initial value of the step size H. Always nonnegative. Default: 0.001|*tend* − *t*0|.
2	HMIN	Minimum value of the step size H. Default: 0.0.
3	HMAX	Maximum value of the step size H. Default: No limit, beyond the machine scale, is imposed on the step size.
4	MXSTEP	Maximum number of steps allowed. Default: 500.
5	MXFCN	Maximum number of function evaluations allowed. Default: No enforced limit.
6	MAXORD	Maximum order of the method. Default: If Adams-Moulton method is used, then 12. If Gear's or BDF method is used, then 5. The defaults are the maximum values allowed.
7	INTRP1	If this value is set nonzero, the subroutine will return before every step with IDO = 4. See Comment 5. Default: 0.
8	INTRP2	If this value is nonzero, the subroutine will return after every successful step with IDO = 5 and return with IDO = 6 after every unsuccessful step. See Comment 5. Default: 0
9	SCALE	A measure of the scale of the problem, such as an approximation to the average value of a norm of the Jacobian along the solution. Default: 1.0
10	INORM	Switch determining error norm. In the following, *e*i is the absolute value of an estimate of the error in *y*i(*t*). Default: 0.0 — min(absolute error, relative error) = max(*e*i/*w*i); *i* = 1, …, *N*, where *w*i = max(|*y*i(*t*)|, 1.0).1 — absolute error = max(*e*i), *i* = 1 …, *N*.2 — max(*e*i / *w*i), *i* = 1 …, *N* where *w*i = max(|*y*i(*t*)|, FLOOR), and FLOOR is the value PARAM(11).3 — Scaled Euclidean norm defined as where *w*i = max(|*y*i(*t*)|, 1.0). Other definitions of YMAX can be specified by the user, as explained in Comment 1.
11	FLOOR	Used in the norm computation associated the parameter INORM. Default: 1.0.
12	METH	Integration method indicator.1 = METH selects the Adams-Moulton method.2 = METH selects Gear's BDF method.Default: 1.

13 MITER Nonlinear solver method indicator.*Note:* If the problem is stiff and a chord or modified Newton method is most efficient, use MITER = 1 or = 2.0 = MITER selects functional iteration. The value IATYPE must be set to zero with this option.1 = MITER selects a chord method with a user-provided Jacobian.2 = MITER selects a chord method with a divided-difference Jacobian.3 = MITER selects a chord method with the Jacobian replaced by a diagonal matrix based on a directional derivative. The value IATYPE must be set to zero with this option.Default: 0.
14 MTYPE Matrix type for *A* (if used) and the Jacobian (if MITER = 1 or = 2). When both are used, *A* and the Jacobian must be of the same type.0 = MTYPE selects full matrices.1 = MTYPE selects banded matrices.2 = MTYPE selects symmetric positive definite matrices.3 = MTYPE selects banded symmetric positive definite matrices.Default: 0.
15 NLC Number of lower codiagonals, used if MTYPE = 1.Default: 0.
16 NUC Number of upper codiagonals, used if MTYPE = 1 or MTYPE = 3.Default: 0.
17 Not used.
18 EPSJ Relative tolerance used in computing divided difference Jacobians.Default: SQRT(AMACH(4)) .
19 IATYPE Type of the matrix *A*.0 = IATYPE implies *A* is not used (the system is explicit).1 = IATYPE if *A* is a constant matrix.2 = IATYPE if *A* depends on *t*.Default: 0.
20 LDA Leading dimension of array *A* exactly as specified in the dimension statement in the calling program. Used if IATYPE is not zero.Default:N if MTYPE = 0 or = 2
NUC + NLC + 1 if MTYPE = 1
NUC + 1 if MTYPE = 3
21–30 Not used.

The following entries in the array PARAM are set by the program:

	PARAM	Meaning
31	HTRIAL	Current trial step size.
32	HMINC	Computed minimum step size.
33	HMAXC	Computed maximum step size.
34	NSTEP	Number of steps taken.
35	NFCN	Number of function evaluations used.
36	NJE	Number of Jacobian evaluations.
37–50		Not used.

Y — Array of size N of dependent variables, *y*(*t*). (Input/Output)
On input, Y contains the initial values, *y*(*t*0). On output, Y contains the approximate solution, *y*(*t*).

Comments

1. Automatic workspace usage is

IVPAG 4N + NMETH + NPW + NIPVT, or
DIVPAG 8N + 2 * NMETH + 2 * NPW + NIPVT units.

Here,

NMETH = 13N if METH is 1,

NMETH = 6N if METH is 2.

NPW = 2N + NPWM + NPWA

where

NPWM = 0 if MITER is 0 or 3,

NPWM = N2 if MITER is 1 or 2, and if MTYPE is 0 or 2.

NPWM = N(2 * NLC + NUC + 1) if MITER is 1 or 2 and MTYPE = 1.

NPWM = N(NLC + 1) if MITER is 1 or 2 and if MTYPE = 3.

NPWA = 0 if IATYPE is 0.

NPWA = N2 if IATYPE is nonzero and MTYPE = 0,

NPWA = N(2 * NLC + NUC + 1) if IATYPE is nonzero and MTYPE = 1

NIPVT = N if MITER is 1 or 2 and MTYPE is 0 or 1,

NIPVT = 1, otherwise.

Workspace and a user-supplied error norm subroutine may be explicitly provided, if desired, by use of I2PAG/DI2PAG. The reference is

```
CALL I2PAG (IDO, N, FCN, FCNJ, A, T, TEND, TOL,
            PARAM, Y, YTEMP, YMAX, ERROR, SAVE1,
            SAVE2, PW, IPVT, VNORM)
```

None of the additional array arguments should be changed from the first call with IDO = 1 until after the final call with IDO = 3. The additional arguments are as follows:

YTEMP — Array of size NMETH. (Workspace)

YMAX — Array of size N containing the maximum Y-values computed so far. (Output)

ERROR — Array of size N containing error estimates for each component of Y. (Output)

SAVE1 — Array of size N. (Workspace)

SAVE2 — Array of size N. (Workspace)

PW — Array of size NPW. PW is used both to store the Jacobian and as workspace. (Workspace)

IPVT — Array of size N. (Workspace)

VNORM — A Fortran SUBROUTINE to compute the norm of the error. (Input)
The routine may be provided by the user, or the IMSL routine I3PRK/DI3PRK may be used. In either case, the name must be declared in a Fortran ENTERNAL statement. If usage of the IMSL routine is intended, then the name I3PRK/DI3PRK should be specified. The usage of the error norm routine is CALL VNORM (N, V, Y, YMAX, ENORM), where

Arg. Definition

N Number of equations. (Input)

V Array of size N containing the vector whose norm is to be computed. (Input)

Y Array of size N containing the values of the dependent variable. (Input)

YMAX Array of size N containing the maximum values of $|y(t)|$. (Input)

ENORM Norm of the vector V. (Output)

VNORM must be declared EXTERNAL in the calling program.

2. Informational errors

Type	Code	
4	1	After some initial success, the integration was halted by repeated error-test failures.
4	2	The maximum number of function evaluations have been used.

4 3 The maximum number of steps allowed have been used. The problem may be stiff.

4 4 On the next step T + H will equal T. Either TOL is too small, or the problem is stiff.
Note: If the Adams-Moulton method is the one used in the integration, then users can switch to the BDF methods. If the BDF methods are being used, then these comments are gratuitous and indicate that the problem is too stiff for this combination of method and value of TOL.

4 5 After some initial success, the integration was halted by a test on TOL.

4 6 Integration was halted after failing to pass the error test even after dividing the initial step size by a factor of 1.0E + 10. The value TOL may be too small.

4 7 Integration was halted after failing to achieve corrector convergence even after dividing the initial step size by a factor of 1.0E + 10. The value TOL may be too small.

4 8 IATYPE is nonzero and the input matrix *A* multiplying *y*′ is singular.

3. Both explicit systems, of the form $y' = f(t, y)$, and implicit systems, $Ay' = f(t, y)$, can be solved. If the system is explicit, then PARAM(19) = 0; and the matrix *A* is not referenced. If the system is implicit, then PARAM(14) determines the data structure of the array A. If PARAM(19) = 1, then A is assumed to be a constant matrix. The value of A used on the first call (with IDO = 1) is saved until after a call with IDO = 3. The value of A must not be changed between these calls. If PARAM(19) = 2, then the matrix is assumed to be a function of *t*.

4. If MTYPE is greater than zero, then MITER must equal 1 or 2.

5. If PARAM(7) is nonzero, the subroutine returns with IDO= 4 and will resume calculation at the point of interruption if re-entered with IDO = 4. If PARAM(8) is nonzero, the subroutine will interrupt immediately after decides to accept the result of the most recent trial step. The value IDO = 5 is returned if the routine plans to accept, or IDO = 6 if it plans to reject. The value IDO may be changed by the user (by changing IDO from 6 to 5) to force acceptance of a step that would otherwise be rejected. Relevant parameters to observe after return from an interrupt are IDO, HTRIAL, NSTEP, NFCN, NJE, T and Y. The array Y contains the newly computed trial value $y(t)$.

Algorithm

The routine IVPAG solves a system of first-order ordinary differential equations of the form $y' = f(t, y)$ or $Ay' = f(t, y)$ with initial conditions where *A* is a square nonsingular matrix of order *N*. Two classes of implicit linear multistep methods are available. The first is the implicit Adams-Moulton method (up to order twelve); the second uses the backward differentiation formulas BDF (up to order five). The BDF method is often called Gear's stiff method. In both cases, because basic formulas are implicit, a system of nonlinear equations must be solved at each step. The deriviative matrix in this system has the form $L = A + \eta J$ where η is a small number computed by IVPAG and *J* is the Jacobian. When it is used, this matrix is computed in the user-supplied routine FCNJ or else it is approximated by divided differences as a default. Using defaults, *A* is the identity matrix. The data structure for the matrix *L* may be identified to be real general, real banded, symmetric positive definite, or banded symmetric positive definite. The default structure for *L* is real general.

Example 1

Euler's equation for the motion of a rigid body not subject to external forces is

$$y_1' = y_2 y_3 \qquad y_1(0) = 0$$
$$y_2' = -y_1 y_3 \qquad y_2(0) = 1$$
$$y_3' = -0.51 y_1 y_2 \qquad y_3(0) = 1$$

Its solution is, in terms of Jacobi elliptic functions, $y1(t)$ = sn(t; k), $y2(t)$ = cn(t; k), $y3(t)$ = dn(t; k) where $k2 = 0.51$. The Adams-Moulton method of `IVPAG` is used to solve this system, since this is the default. All parameters are set to defaults.

The last call to `IVPAG` with `IDO` = 3 releases IMSL workspace that was reserved on the first call to `IVPAG`. It is not necessary to release the workspace in this example because the program ends after solving a single problem. The call to release workspace is made as a model of what would be needed if the program included further calls to IMSL routines.

Because `PARAM`(13) = `MITER` = 0, functional iteration is used and so subroutine `FCNJ` is never called. It is included only because the calling sequence for `IVPAG` requires it.

```
      INTEGER    N, NPARAM
      PARAMETER  (N=3, NPARAM=50)
C                                SPECIFICATIONS FOR LOCAL VARIABLES
      INTEGER    IDO, IEND, NOUT
      REAL       A(1,1), PARAM(NPARAM), T, TEND, TOL, Y(N)
C                                SPECIFICATIONS FOR SUBROUTINES
      EXTERNAL   IVPAG, SSET, UMACH
C                                SPECIFICATIONS FOR FUNCTIONS
      EXTERNAL   FCN, FCNJ
C                                Initialize
      CALL SSET (NPARAM, 0.0, PARAM, 1)
C
      IDO  = 1
      T    = 0.0
      Y(1) = 0.0
      Y(2) = 1.0
      Y(3) = 1.0
      TOL  = 1.0E-6
C                                Write title
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99998)
C                                Integrate ODE
      IEND = 0
   10 CONTINUE
      IEND = IEND + 1
      TEND = IEND
C                                The array a(*,*) is not used.
      CALL IVPAG (IDO, N, FCN, FCNJ, A, T, TEND, TOL, PARAM, Y)
      IF (IEND .LE. 10) THEN
         WRITE (NOUT,99999) T, Y
C                                Finish up
         IF (IEND .EQ. 10) IDO = 3
         GO TO 10
      END IF
99998 FORMAT (11X, 'T', 14X, 'Y(1)', 11X, 'Y(2)', 11X, 'Y(3)')
99999 FORMAT (4F15.5)
      END
C
      SUBROUTINE FCN (N, X, Y, YPRIME)
C                                SPECIFICATIONS FOR ARGUMENTS
      INTEGER    N
      REAL       X, Y(N), YPRIME(N)
```

```
C
      YPRIME(1) = Y(2)*Y(3)
      YPRIME(2) = -Y(1)*Y(3)
      YPRIME(3) = -0.51*Y(1)*Y(2)
      RETURN
      END
C
      SUBROUTINE FCNJ (N, X, Y, DYPDY)
C                                  SPECIFICATIONS FOR ARGUMENTS
      INTEGER    N
      REAL       X, Y(N), DYPDY(N,*)
C                                  This subroutine is never called
      RETURN
      END
```

Output

```
     T              Y(1)           Y(2)           Y(3)
 1.00000         0.80220         0.59705        0.81963
 2.00000         0.99537        -0.09615        0.70336
 3.00000         0.64141        -0.76720        0.88892
 4.00000        -0.26961        -0.96296        0.98129
 5.00000        -0.91173        -0.41079        0.75899
 6.00000        -0.95751         0.28841        0.72967
 7.00000        -0.42877         0.90342        0.95197
 8.00000         0.51092         0.85963        0.93106
 9.00000         0.97567         0.21926        0.71730
10.00000         0.87790        -0.47884        0.77906
```

Example 2

The BDF method of IVPAG is used to solve Example 2 of IVPRK. We set PARAM(12) = 2 to designate the BDF method. A chord or modified Newton method, with the Jacobian computed by divided differences, is used to solve the nonlinear equations. Thus, we set PARAM(13) = 2. The number of evaluations of y' is printed after the last output point, showing the efficiency gained when using a stiff solver compared to using IVPRK on this problem. The number of evaluations may vary, depending on the accuracy and other arithmetic characteristics of the computer.

```
      INTEGER    MXPARM, N
      PARAMETER  (MXPARM=50, N=2)
C                                  SPECIFICATIONS FOR PARAMETERS
      INTEGER    MABSE, MBDF, MSOLVE
      PARAMETER  (MABSE=1, MBDF=2, MSOLVE=2)
C                                  SPECIFICATIONS FOR LOCAL VARIABLES
      INTEGER    IDO, ISTEP, NOUT
      REAL       A(1,1), PARAM(MXPARM), T, TEND, TOL, Y(N)
C                                  SPECIFICATIONS FOR SUBROUTINES
      EXTERNAL   IVPAG, SSET, UMACH
C                                  SPECIFICATIONS FOR FUNCTIONS
      EXTERNAL   FCN, FCNJ
C
      CALL UMACH (2, NOUT)
C                                  Set initial conditions
      T = 0.0
      Y(1) = 1.0
      Y(2) = 0.0
```

```
C                                  Set error tolerance
      TOL = 0.001
C                                  Set PARAM to defaults
      CALL SSET (MXPARM, 0.0, PARAM, 1)
C                                  Select absolute error control
      PARAM(10) = MABSE
C                                  Select BDF method
      PARAM(12) = MBDF
C                                  Select chord method and
C                                  a divided difference Jacobian.
      PARAM(13) = MSOLVE
C                                  Print header
      WRITE (NOUT,99998)
      IDO = 1
      ISTEP = 0
   10 CONTINUE
      ISTEP = ISTEP + 24
      TEND = ISTEP
C                                  The array a(*,*) is not used.
      CALL IVPAG (IDO, N, FCN, FCNJ, A, T, TEND, TOL, PARAM, Y)
      IF (ISTEP .LE. 240) THEN
         WRITE (NOUT,'(I6,3F12.3)') ISTEP/24, T, Y
C                                  Final call to release workspace
         IF (ISTEP .EQ. 240) IDO = 3
         GO TO 10
      END IF
C                                  Show number of function calls.
      WRITE (NOUT,99999) PARAM(35)
99998 FORMAT (4X, 'ISTEP', 5X, 'Time', 9X, 'Y1', 11X, 'Y2')
99999 FORMAT (4X, 'Number of fcn calls with IVPAG =', F6.0)
      END
      SUBROUTINE FCN (N, T, Y, YPRIME)
C                                  SPECIFICATIONS FOR ARGUMENTS
      INTEGER    N
      REAL       T, Y(N), YPRIME(N)
C                                  SPECIFICATIONS FOR SAVE VARIABLES
      REAL       AK1, AK2, AK3
      SAVE       AK1, AK2, AK3
C
      DATA AK1, AK2, AK3/294.0E0, 3.0E0, 0.01020408E0/
C
      YPRIME(1) = -Y(1) - Y(1)*Y(2) + AK1*Y(2)
      YPRIME(2) = -AK2*Y(2) + AK3*(1.0E0-Y(2))*Y(1)
      RETURN
      END
      SUBROUTINE FCNJ (N, T, Y, DYPDY)
C                                  SPECIFICATIONS FOR ARGUMENTS
      INTEGER    N
      REAL       T, Y(N), DYPDY(N,*)
C
      RETURN
      END
```

Output

```
ISTEP     Time          Y1          Y2
 1      24.000       0.689       0.002
 2      48.000       0.636       0.002
```

```
 3      72.000       0.590       0.002
 4      96.000       0.550       0.002
 5     120.000       0.515       0.002
 6     144.000       0.485       0.002
 7     168.000       0.458       0.002
 8     192.000       0.434       0.001
 9     216.000       0.412       0.001
10     240.000       0.392       0.001
Number of fcn calls with IVPAG =   73.
```

Example 3

The BDF method of IVPAG is used to solve the so-called Robertson problem:

$$y_1' = -c_1 y_1 + c_2 y_2 y_3 \qquad y_1(0) = 1$$

$$y_2' = -y_1' - y_3' \qquad y_2(0) = 0$$

$$y_3' = c_3 y_2^2 \qquad y_3(0) = 0$$

$$c_1 = 0.04,\ c_2 = 10^4,\ c_3 = 3 \times 10^7 \qquad 0 \le t \le 10$$

Output is obtained after each unit of the independent variable. A user-provided subroutine for the Jacobian matrix is used. An absolute error tolerance of 10-5 is required.

```
      INTEGER    MXPARM, N
      PARAMETER  (MXPARM=50, N=3)
C                                SPECIFICATIONS FOR PARAMETERS
      INTEGER    MABSE, MBDF, MSOLVE
      PARAMETER  (MABSE=1, MBDF=2, MSOLVE=1)
C                                SPECIFICATIONS FOR LOCAL VARIABLES
      INTEGER    IDO, ISTEP, NOUT
      REAL       A(1,1), PARAM(MXPARM), T, TEND, TOL, Y(N)
C                                SPECIFICATIONS FOR SUBROUTINES
      EXTERNAL   IVPAG, SSET, UMACH
C                                SPECIFICATIONS FOR FUNCTIONS
      EXTERNAL   FCN, FCNJ
C
      CALL UMACH (2, NOUT)
C                                Set initial conditions
      T = 0.0
      Y(1) = 1.0
      Y(2) = 0.0
      Y(3) = 0.0
C                                Set error tolerance
      TOL = 1.0E-5
C                                Set PARAM to defaults
      CALL SSET (MXPARM, 0.0, PARAM, 1)
C                                Select absolute error control
      PARAM(10) = MABSE
C                                Select BDF method
      PARAM(12) = MBDF
C                                Select chord method and
C                                a user-provided Jacobian.
      PARAM(13) = MSOLVE
C                                Print header
      WRITE (NOUT,99998)
      IDO = 1
      ISTEP = 0
   10 CONTINUE
```

```
      ISTEP = ISTEP + 1
      TEND = ISTEP
C                                  The array a(*,*) is not used.
      CALL IVPAG (IDO, N, FCN, FCNJ, A, T, TEND, TOL, PARAM, Y)
      IF (ISTEP .LE. 10) THEN
         WRITE (NOUT,'(I6,F12.2,3F13.5)') ISTEP, T, Y
C                                  Final call to release workspace
         IF (ISTEP .EQ. 10) IDO = 3
         GO TO 10
      END IF
99998 FORMAT (4X, 'ISTEP', 5X, 'Time', 9X, 'Y1', 11X, 'Y2', 11X,
     &       'Y3')
      END
      SUBROUTINE FCN (N, T, Y, YPRIME)
C                                  SPECIFICATIONS FOR ARGUMENTS
      INTEGER    N
      REAL       T, Y(N), YPRIME(N)
C                                  SPECIFICATIONS FOR SAVE VARIABLES
      REAL       C1, C2, C3
      SAVE       C1, C2, C3
C
      DATA C1, C2, C3/0.04E0, 1.0E4, 3.0E7/
C
      YPRIME(1) = -C1*Y(1) + C2*Y(2)*Y(3)
      YPRIME(3) = C3*Y(2)**2
      YPRIME(2) = -YPRIME(1) - YPRIME(3)
      RETURN
      END
      SUBROUTINE FCNJ (N, T, Y, DYPDY)
C                                  SPECIFICATIONS FOR ARGUMENTS
      INTEGER    N
      REAL       T, Y(N), DYPDY(N,*)
C                                  SPECIFICATIONS FOR SAVE VARIABLES
      REAL       C1, C2, C3
      SAVE       C1, C2, C3
C                                  SPECIFICATIONS FOR SUBROUTINES
      EXTERNAL   SSET
C
      DATA C1, C2, C3/0.04E0, 1.0E4, 3.0E7/
C                                  Clear array to zero
      CALL SSET (N**2, 0.0, DYPDY, 1)
C                                  Compute partials
      DYPDY(1,1) = -C1
      DYPDY(1,2) = C2*Y(3)
      DYPDY(1,3) = C2*Y(2)
      DYPDY(3,2) = 2.0*C3*Y(2)
      DYPDY(2,1) = -DYPDY(1,1)
      DYPDY(2,2) = -DYPDY(1,2) - DYPDY(3,2)
      DYPDY(2,3) = -DYPDY(1,3)
      RETURN
      END
```

Output

```
 ISTEP     Time         Y1           Y2           Y3
 1         1.00      0.96647      0.00003      0.03350
 2         2.00      0.94164      0.00003      0.05834
 3         3.00      0.92191      0.00002      0.07806
```

```
 4        4.00      0.90555      0.00002      0.09443
 5        5.00      0.89153      0.00002      0.10845
 6        6.00      0.87928      0.00002      0.12070
 7        7.00      0.86838      0.00002      0.13160
 8        8.00      0.85855      0.00002      0.14143
 9        9.00      0.84959      0.00002      0.15039
10       10.00      0.84136      0.00002      0.15862
```

Example 4

Solve the partial differential equation

$$e^{-t}\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

with the initial condition

$u(t = 0, x) = \sin x$

and the boundary conditions

$u(t, x = 0) = u(t, x = \pi) = 0$

on the square $[0, 1] \times [0, \pi]$, using the method of lines with a piecewise-linear Galerkin discretization. The exact solution is $u(t, x) = \exp(1 - et) \sin x$. The interval $[0, \pi]$ is divided into equal intervals by choosing breakpoints $x_k = k\pi/(N + 1)$ for $k = 0, \ldots, N + 1$. The unknown function $u(t, x)$ is approximated by

$$\sum_{k=1}^{N} c_k(t)\phi_k(x)$$

where $\phi_k(x)$ is the piecewiselinear function that equals 1 at x_k and is zero at all of the other breakpoints. We approximate the partial differential equation by a system of N ordinary differential equations, $A\,dc/dt = Rc$ where A and R are matrices of order N. The matrix A is given by

$$A_{ij} = e^{-t}\int_0^{\pi} \phi_i(x)\phi_j(x)\,dx = \begin{cases} e^{-t}2h/3 & \text{if } i = j \\ e^{-t}h/6 & \text{if } i = j \pm 1 \\ 0 & \text{otherwise} \end{cases}$$

where $h = 1/(N + 1)$ is the mesh spacing. The matrix R is given by

$$R_{ij} = \int_0^{\pi} \phi_i''(x)\phi_j(x)\,dx = -\int_0^{\pi} \phi_i'(x)\phi_j'(x)\,dx = \begin{cases} -2/h & \text{if } i = j \\ 1/h & \text{if } i = j \pm 1 \\ 0 & \text{otherwise} \end{cases}$$

The integrals involving

ϕ_i''

are assigned the values of the integrals on the right-hand side, by using the boundary values and integration by parts. Because this system may be stiff, Gear's BDF method is used.

In the following program, the array `Y(1:N)` corresponds to the vector of coefficients, c. Note that `Y` contains `N + 2` elements; `Y(0)` and `Y(N + 1)` are used to store the boundary values. The matrix A depends on t so we set `PARAM(19) = 2` and evaluate A when `IVPAG` returns with `IDO = 7`. The subroutine `FCN` computes the vector Rc, and the subroutine `FCNJ` computes R. The matrices A and R are stored as band-symmetric positive-definite structures having one upper co-diagonal.

```
      INTEGER    LDA, N, NPARAM, NUC
      PARAMETER  (N=9, NPARAM=50, NUC=1, LDA=NUC+1)
C                                SPECIFICATIONS FOR PARAMETERS
      INTEGER    NSTEP
      PARAMETER  (NSTEP=4)
C                                SPECIFICATIONS FOR LOCAL VARIABLES
      INTEGER    I, IATYPE, IDO, IMETH, INORM, ISTEP, MITER, MTYPE
      REAL       A(LDA,N), C, HINIT, PARAM(NPARAM), PI, T, TEND,
TMAX,
     &           TOL, XPOINT(0:N+1), Y(0:N+1)
      CHARACTER  TITLE*10
C                                SPECIFICATIONS FOR COMMON /COMHX/
      COMMON     /COMHX/ HX
      REAL       HX
C                                SPECIFICATIONS FOR INTRINSICS
      INTRINSIC  EXP, REAL, SIN
      REAL       EXP, REAL, SIN
C                                SPECIFICATIONS FOR SUBROUTINES
      EXTERNAL   IVPAG, SSET, WRRRN
C                                SPECIFICATIONS FOR FUNCTIONS
      EXTERNAL   CONST, FCN, FCNJ
      REAL       CONST
C                                Initialize PARAM
      HINIT  = 1.0E-3
      INORM  = 1
      IMETH  = 2
      MITER  = 1
      MTYPE  = 3
      IATYPE = 2
      CALL SSET (NPARAM, 0.0, PARAM, 1)
      PARAM(1)  = HINIT
      PARAM(10) = INORM
      PARAM(12) = IMETH
      PARAM(13) = MITER
      PARAM(14) = MTYPE
      PARAM(16) = NUC
      PARAM(19) = IATYPE
C                                Initialize other arguments
      PI = CONST('PI')
      HX = PI/REAL(N+1)
      CALL SSET (N-1, HX/6., A(1,2), LDA)
      CALL SSET (N, 2.*HX/3., A(2,1), LDA)
      DO 10  I=0, N + 1
         XPOINT(I) = I*HX
         Y(I)      = SIN(XPOINT(I))
   10 CONTINUE
      TOL  = 1.0E-6
      T    = 0.0
      TMAX = 1.0
C                                Integrate ODE
      IDO   = 1
      ISTEP = 0
   20 CONTINUE
      ISTEP = ISTEP + 1
      TEND  = TMAX*REAL(ISTEP)/REAL(NSTEP)
   30 CALL IVPAG (IDO, N, FCN, FCNJ, A, T, TEND, TOL, PARAM, Y(1))
C                                Set matrix A
      IF (IDO .EQ. 7) THEN
```

```
         C = EXP(-T)
         CALL SSET (N-1, C*HX/6., A(1,2), LDA)
         CALL SSET (N, 2.*C*HX/3., A(2,1), LDA)
         GO TO 30
      END IF
      IF (ISTEP .LE. NSTEP) THEN
C                                  Print solution
         WRITE (TITLE,'(A,F5.3,A)') 'U(T=', T, ')'
         CALL WRRRN (TITLE, 1, N+2, Y, 1, 0)
C                                  Final call to release workspace
         IF (ISTEP .EQ. NSTEP) IDO = 3
         GO TO 20
       END IF
       END
C
      SUBROUTINE FCN (N, T, Y, YPRIME)
C                                  SPECIFICATIONS FOR ARGUMENTS
      INTEGER    N
      REAL       T, Y(*), YPRIME(N)
C                                  SPECIFICATIONS FOR LOCAL VARIABLES
      INTEGER    I
C                                  SPECIFICATIONS FOR COMMON /COMHX/
      COMMON     /COMHX/ HX
      REAL       HX
C                                  SPECIFICATIONS FOR SUBROUTINES
      EXTERNAL   SSCAL
C
      YPRIME(1) = -2.0*Y(1) + Y(2)
      DO 10  I=2, N - 1
         YPRIME(I) = -2.0*Y(I) + Y(I-1) + Y(I+1)
   10 CONTINUE
      YPRIME(N) = -2.0*Y(N) + Y(N-1)
      CALL SSCAL (N, 1.0/HX, YPRIME, 1)
      RETURN
      END
C
      SUBROUTINE FCNJ (N, T, Y, DYPDY)
C                                  SPECIFICATIONS FOR ARGUMENTS
      INTEGER    N
      REAL       T, Y(*), DYPDY(2,*)
C                                  SPECIFICATIONS FOR COMMON /COMHX/
      COMMON     /COMHX/ HX
      REAL       HX
C                                  SPECIFICATIONS FOR SUBROUTINES
      EXTERNAL   SSET
C
      CALL SSET (N-1, 1.0/HX, DYPDY(1,2), 2)
      CALL SSET (N, -2.0/HX, DYPDY(2,1), 2)
      RETURN
      END
```

Output

```
                          U(T=0.250)
     1        2        3        4        5        6        7
8
0.0000   0.2321   0.4414   0.6076   0.7142   0.7510   0.7142
0.6076
```

```
     9       10       11
0.4414   0.2321   0.0000


                          U(T=0.500)
     1        2        3        4        5        6        7
8
0.0000   0.1607   0.3056   0.4206   0.4945   0.5199   0.4945
0.4206


     9       10       11
0.3056   0.1607   0.0000


                          U(T=0.750)
     1        2        3        4        5        6        7
8
0.0000   0.1002   0.1906   0.2623   0.3084   0.3243   0.3084
0.2623


     9       10       11
0.1906   0.1002   0.0000


                          U(T=1.000)
     1        2        3        4        5        6        7
8
0.0000   0.0546   0.1039   0.1431   0.1682   0.1768   0.1682
0.1431


     9       10       11
0.1039   0.0546   0.0000
```