NAG Toolbox for MATLAB

d02nn

1 Purpose

d02nn is a reverse communication function for integrating stiff systems of implicit ordinary differential equations coupled with algebraic equations.

2 Syntax

```
[t, tout, y, ydot, rwork, inform, ysav, wkjac, jacpvt, imon, inln, ires,
irevcm, lderiv, ifail] = d02nn(neq, t, tout, y, ydot, rwork, rtol, atol,
itol, inform, ysav, wkjac, jacpvt, imon, inln, ires, irevcm, lderiv,
itask, itrace, 'sdysav', sdysav, 'nwkjac', nwkjac, 'njcpvt', njcpvt)
```

3 Description

d02nn is a general purpose function for integrating the initial value problem for a stiff system of implicit ordinary differential equations coupled with algebraic equations, written in the form

$$A(t,y)y' = g(t,y).$$

An outline of a typical calling program is given below:

```
%
% data initialisation
%
call linear algebra setup routine
call integrator setup routine
irevcm = int32(0);
[..., irevcm, ...] = d02nn();
while (irevcm > 0)
  if (irevcm > 7 and irevcm < 11)
    if (irevcm == 8)
      supply the jacobian matrix                                  (i)
    elseif (irevcm == 9)
      perform monitoring tasks requested by the user             (ii)
    elseif (irevcm == 10)
      indicates an unsuccessful step
             end          else                evaluate   the   residual
(iii)
  end
  [..., irevcm, ...] = d02nn();
end
%
% post processing (optional linear algebra diagnostic call
% (sparse case only), optional integrator diagnostic call)
%
```

There are three major operations that may be required of the calling (sub)program on an intermediate return (**irevcm** $\neq 0$) from d02nn; these are denoted (i), (ii) and (iii) above.

The following sections describe in greater detail exactly what is required of each of these operations.

(i) Supply the Jacobian matrix

You need only provide this facility if the parameter **jceval** = 'A' (or 'F' if using sparse matrix linear algebra) in a call to the linear algebra setup function. If the Jacobian matrix is to be evaluated numerically by the integrator, then the remainder of section (i) can be ignored.

We must define the system of nonlinear equations which is solved internally by the integrator. The time derivative, y', has the form

$$y' = (y - z)/(hd),$$

where h is the current step size and d is a parameter that depends on the integration method in use. The vector y is the current solution and the vector z depends on information from previous time steps. This means that $\frac{d}{dy'}(\ ) = \frac{1}{(hd)}\frac{d}{dy}(\ )$.

The system of nonlinear equations that is solved has the form

$$A(t,y)y' - g(t,y) = 0$$

but is solved in the form

$$r(t,y) = 0,$$

where r is the function defined by

$$r(t,y) = (hd)(A(t,y)(y-z)/(hd) - g(t,y)).$$

It is the Jacobian matrix $\dfrac{\partial r}{\partial y}$ that you must supply as follows:

$$\frac{\partial r_i}{\partial y_j} = a_{ij}(t,y) + hd\frac{\partial}{\partial y_j}\left(\sum_{k=1}^{\mathbf{neq}} a_{ik}(t,y){y'}_k - g_i(t,y)\right),$$

where t, h and d are located in **rwork**(19), **rwork**(16) and **rwork**(20) respectively and the arrays **y** and **ydot** contain the current solution and time derivatives respectively. Only the nonzero elements of the Jacobian need be set, since the locations where it is to be stored are preset to zero.

Hereafter in this document this operation will be referred to as JAC.

(ii) Perform tasks requested by you

This operation is essentially a monitoring function and additionally provides the opportunity of changing the current values of **y**, **ydot**, HNEXT (the step size that the integrator proposes to take on the next step), HMIN (the minimum step size to be taken on the next step), and HMAX (the maximum step size to be taken on the next step). The scaled local error at the end of a time step may be obtained by calling the double function d02za as follows:

```
[errloc, ifail] = d02za(rwork(51+neqmax:51+neqmax+neq-1), rwork(51:51+neq-
1));
% Check ifail before proceeding
```

The following gives details of the location within the array **rwork** of variables that may be of interest to you:

Variable	Specification	Location
TCURR	the current value of the independent variable	**rwork**(19)
HLAST	last step size successfully used by the integrator	**rwork**(15)
HNEXT	step size that the integrator proposes to take on the next step	**rwork**(16)
HMIN	minimum step size to be taken on the next step	**rwork**(17)
HMAX	maximum step size to be taken on the next step	**rwork**(18)
NQU	the order of the integrator used on the last step	**rwork**(10)

You are advised to consult the description of (sub)program **monitr** in d02ng for details on what optional input can be made.

If either **y** or **ydot** are changed, then **imon** must be set to 2 before return to d02nn. If either of the values HMIN or HMAX are changed, then **imon** must be set ≥ 3 before return to d02nn. If HNEXT is changed, then **imon** must be set to 4 before return to d02nn.

In addition you can force d02nn to evaluate the residual vector

$$A(t,y)y' - g(t,y)$$

by setting **imon** $= 0$ and **inln** $= 3$ and then returning to d02nn; on return to this monitoring operation the residual vector will be stored in **rwork**$(50 + 2 \times$ **ldysav** $+ i)$, for $i = 1, 2, \ldots,$ **neq**.

Hereafter in this document this operation will be referred to as (sub)program monitr.

(iii) Evaluate the residual

This operation must evaluate the residual

$$r = g(t,y) - A(t,y)y' \tag{1}$$

in one case and

$$r = -A(t,y)y' \tag{2}$$

in another, where t is located in **rwork**(19). The form of the residual that is returned is determined by the value of **ires** returned by d02nn. If **ires** $= -1$, then the residual defined by equation (2) above must be returned; if **ires** $= 1$, then the residual returned by equation (1) above must be returned.

Hereafter in this document this operation will be referred to as user-supplied (sub)program resid.

4 References

See the D02M/N Sub-chapter Introduction.

5 Parameters

Note: this function uses **reverse communication.** Its use involves an initial entry, intermediate exits and re-entries, and a final exit, as indicated by the **parameter IREVCM**. Between intermediate exits and re-entries, **all parameters other than ydot, rwork, wkjac, imon, inln and ires must remain unchanged**.

5.1 Compulsory Input Parameters

1: **neq – int32 scalar**

On initial entry: the number of equations to be solved.

Constraint: **neq** ≥ 1.

2: **t – double scalar**

On initial entry: the value of the independent variable t. The input value of **t** is used only on the first call as the initial point of the integration.

3: **tout – double scalar**

On initial entry: the next value of t at which a computed solution is desired. For the initial t, the input value of **tout** is used to determine the direction of integration. Integration is permitted in either direction (see also **itask**).

Constraint: **tout** $\neq$ **t**.

4: **y**(**ldysav**) **– double array**

ldysav, the first dimension of the array, must be at least **neq**.

On initial entry: the values of the dependent variables (solution). On the first call the first **neq** elements of y must contain the vector of initial values.

5: **ydot**(**ldysav**) **– double array**

ldysav, the first dimension of the array, must be at least **neq**.

On initial entry: if **lderiv**(1) $=$ **true**, **ydot** must contain approximations to the time derivatives y' of the vector y. If **lderiv**(1) $=$ **false**, then **ydot** need not be set on entry.

6: **rwork**(**50** $+$ **4** $\times$ **ldysav**) **– double array**

On initial entry: must be the same array as used by one of the method setup functions d02nv, d02nw or d02mv, and by one of the storage setup functions d02nv, d02nt or d02nu. The contents of **rwork** must not be changed between any call to a setup function and the first call to d02nn.

On intermediate re-entry: must contain residual evaluations as described under the parameter **irevcm**.

7: **rtol**$(*)$ – **double array**

Note: the dimension of the array **rtol** must be at least 1 if **itol** $= 1$ or **itol** $= 2$, and at least **neq** otherwise.

On initial entry: the relative local error tolerance.

Constraint: **rtol**$(i) \geq 0.0$ for all relevant i (see **itol**).

8: **atol**$(*)$ – **double array**

Note: the dimension of the array **atol** must be at least 1 if **itol** $= 1$ or **itol** $= 3$, and at least **neq** otherwise.

On initial entry: the absolute local error tolerance.

Constraint: **atol**$(i) \geq 0.0$ for all relevant i (see **itol**).

9: **itol** – **int32 scalar**

On initial entry: a value to indicate the form of the local error test. **itol** indicates to d02nn whether to interpret either or both of **rtol** or **atol** as a vector or a scalar. The error test to be satisfied is $\|e_i/w_i\| < 1.0$, where w_i is defined as follows:

itol	**rtol**	**atol**	w_i
1	scalar	scalar	**rtol**$(1) \times \|y_i\| +$ **atol**(1)
2	scalar	vector	**rtol**$(1) \times \|y_i\| +$ **atol**(i)
3	vector	scalar	**rtol**$(i) \times \|y_i\| +$ **atol**(1)
4	vector	vector	**rtol**$(i) \times \|y_i\| +$ **atol**(i)

e_i is an estimate of the local error in y_i, computed internally, and the choice of norm to be used is defined by a previous call to an integrator setup function.

Constraint: $1 \leq$ **itol** ≤ 4.

10: **inform**(23) – **int32 array**

11: **ysav**(**ldysav**,**sdysav**) – **double array**

ldysav, the first dimension of the array, must be at least **neq**.

On initial entry: An appropriate value for **sdysav** is described in the specifications of the integrator setup functions d02mv, d02nv and d02nw. This value must be the same as that supplied to the integrator setup function.

12: **wkjac**(**nwkjac**) – **double array**

On intermediate re-entry: elements of the Jacobian as defined under the description of **irevcm**. If a numerical Jacobian was requested then **wkjac** is used for workspace.

13: **jacpvt**(**njcpvt**) – **int32 array**

On initial entry: The actual size depends on the linear algebra method used. An appropriate value for **njcpvt** is described in the specifications of the linear algebra setup functions d02nt and d02nu for banded and sparse matrix linear algebra respectively. This value must be the same as that supplied to the linear algebra setup function. When full matrix linear algebra is chosen, the array **jacpvt** is not used and hence **njcpvt** should be set to 1.

14: **imon** – **int32 scalar**

On intermediate re-entry: may be reset to determine subsequent action in d02nn.

imon $= -2$

Integration is to be halted. A return will be made from d02nn to the calling (sub)program with **ifail** $= 12$.

imon $= -1$

Allow d02nn to continue with its own internal strategy. The integrator will try up to three restarts unless **imon** $\neq -1$.

imon $= 0$

Return to the internal nonlinear equation solver, where the action taken is determined by the value of **inln**.

imon $= 1$

Normal exit to d02nn to continue integration.

imon $= 2$

Restart the integration at the current time point. The integrator will restart from order 1 when this option is used. The internal initialization module solves for new values of y and y' by using the values supplied in **y** and **ydot** by the MONITR operation (see Section 3) as initial estimates.

imon $= 3$

Try to continue with the same step size and order as was to be used before entering the MONITR operation (see Section 3). HMIN and HMAX may be altered if desired.

imon $= 4$

Continue the integration but using a new value of HNEXT and possibly new values of HMIN and HMAX.

15: **inln – int32 scalar**

On intermediate re-entry: with **imon** $= 0$ and **irevcm** $= 9$, **inln** specifies the action to be taken by the internal nonlinear equation solver. By setting **inln** $= 3$ and returning to d02nn, the residual vector is evaluated and placed in **rwork**$(50 + 2 \times$ **ldysav** $+ i)$, for $i = 1, 2, \ldots,$ **neq** and then the MONITR operation (see Section 3) is invoked again. At present this is the only option available: **inln** must not be set to any other value.

16: **ires – int32 scalar**

On intermediate re-entry: should be unchanged unless one of the following actions is required of d02nn in which case **ires** should be set accordingly.

ires $= 2$

Indicates to d02nn that control should be passed back immediately to the calling (sub)program with the error indicator set to **ifail** $= 11$.

ires $= 3$

Indicates to d02nn that an error condition has occurred in the solution vector, its time derivative or in the value of t. The integrator will use a smaller time step to try to avoid this condition. If this is not possible d02nn returns to the calling (sub)program with the error indicator set to **ifail** $= 7$.

ires $= 4$

Indicates to d02nn to stop its current operation and to enter the MONITR operation (see Section 3) immediately.

17: **irevcm – int32 scalar**

On initial entry: must contain 0.

On intermediate re-entry: should remain unchanged.

Constraint: $0 \leq$ **irevcm** ≤ 11.

18: **lderiv**(**2**) – **logical array**

On initial entry: **lderiv**(1) must be set to **true** if you have supplied both an initial y and an initial y'. **lderiv**(1) must be set to **false** if only the initial y has been supplied.

lderiv(2) must be set to **true** if the integrator is to use a modified Newton method to evaluate the initial y and y'. Note that y and y', if supplied, are used as initial estimates. This method involves taking a small step at the start of the integration, and if **itask** $= 6$ on entry, **t** and **tout** will be set to the result of taking this small step. **lderiv**(2) must be set to **false** if the integrator is to use functional iteration to evaluate the initial y and y', and if this fails a modified Newton method will then be attempted. **lderiv**$(2) =$ **true** is recommended if there are implicit equations or the initial y and y' are zero.

19: **itask – int32 scalar**

On initial entry: the task to be performed by the integrator.

itask $= 1$

Normal computation of output values of $y(t)$ at $t =$ **tout** (by overshooting and interpolating).

itask $= 2$

Take one step only and return.

itask $= 3$

Stop at the first internal integration point at or beyond $t =$ **tout** and return.

itask $= 4$

Normal computation of output values of $y(t)$ at $t =$ **tout** but without overshooting $t =$ **tcrit**. **tcrit** must be specified as an option in one of the integrator setup functions prior to the first call to the integrator, or specified in the optional input function prior to a continuation call. **tcrit** (e.g., see d02nv) may be equal to or beyond **tout**, but not before it in the direction of integration.

itask $= 5$

Take one step only and return, without passing **tcrit** (e.g., see d02nv). **tcrit** must be specified under **itask** $= 4$.

itask $= 6$

The integrator will solve for the initial values of y and y' only and then return to the calling (sub)program without doing the integration. This option can be used to check the initial values of y and y'. Functional iteration or a 'small' backward Euler method used in conjunction with a damped Newton iteration is used to calculate these values (see **lderiv**). Note that if a backward Euler step is used then the value of t will have been advanced a short distance from the initial point.

Note: if d02nn is recalled with a different value of **itask** (and **tout** altered) then the initialization procedure is repeated, possibly leading to different initial conditions.

Constraint: $1 \leq$ **itask** ≤ 6.

20: **itrace – int32 scalar**

On initial entry: the level of output that is printed by the integrator. **itrace** may take the value -1, 0, 1, 2 or 3.

itrace < -1

-1 is assumed and similarly if **itrace** > 3, then 3 is assumed.

itrace $= -1$

No output is generated.

itrace $= 0$

Only warning messages are printed on the current error message unit (see x04aa).

itrace > 0

Warning messages are printed as above, and on the current advisory message unit (see x04ab) output is generated which details Jacobian entries, the nonlinear iteration and the time integration. The advisory messages are given in greater detail the larger the value of **itrace**.

5.2 Optional Input Parameters

1: **sdysav – int32 scalar**

Default: The second dimension of the array **ysav**.

On initial entry: An appropriate value for **sdysav** is described in the specifications of the integrator setup functions d02mv, d02nv and d02nw. This value must be the same as that supplied to the integrator setup function.

2: **nwkjac – int32 scalar**

Default: The dimension of the array **wkjac**.

On initial entry: The actual size depends on the linear algebra method used. An appropriate value for **nwkjac** is described in the specifications of the linear algebra setup functions d02ns, d02nt and d02nu for full, banded and sparse matrix linear algebra respectively. This value must be the same as that supplied to the linear algebra setup function.

3: **njcpvt – int32 scalar**

Default: The dimension of the array **jacpvt**.

On initial entry: The actual size depends on the linear algebra method used. An appropriate value for **njcpvt** is described in the specifications of the linear algebra setup functions d02nt and d02nu for banded and sparse matrix linear algebra respectively. This value must be the same as that supplied to the linear algebra setup function. When full matrix linear algebra is chosen, the array **jacpvt** is not used and hence **njcpvt** should be set to 1.

5.3 Input Parameters Omitted from the MATLAB Interface

ldysav

5.4 Output Parameters

1: **t – double scalar**

On final exit: the value at which the computed solution y is returned (usually at **tout**).

2: **tout – double scalar**

Is unaltered unless **itask** $= 6$ and **lderiv**(2) $=$ **true** on entry (see also **itask** and **lderiv**) in which case **tout** will be set to the result of taking a small step at the start of the integration.

3: **y**(**ldysav**) **– double array**

On final exit: the computed solution vector evaluated at **t** (usually $t =$ **tout**).

4: **ydot**(**ldysav**) – **double array**

On final exit: contains the time derivatives y' of the vector y at the last integration point.

5: **rwork**(**50** + **4** × **ldysav**) – **double array**

On intermediate exit: contains information for **jac**, **resid** and **monitr** operations as described under Section 3 and the parameter **irevcm**.

6: **inform**(**23**) – **int32 array**

7: **ysav**(**ldysav,sdysav**) – **double array**

8: **wkjac**(**nwkjac**) – **double array**

On intermediate exit: the Jacobian is overwritten.

9: **jacpvt**(**njcpvt**) – **int32 array**

10: **imon** – **int32 scalar**

On intermediate exit: used to pass information between d02nn and the MONITR operation (see Section 3). With **irevcm** = 9, **imon** contains a flag indicating under what circumstances the return from d02nn occurred:

imon = −2

Exit from d02nn after **ires** = 4 (set in the RESID operation (see Section 3) caused an early termination (this facility could be used to locate discontinuities).

imon = −1

The current step failed repeatedly.

imon = 0

Exit from d02nn after a call to the internal nonlinear equation solver.

imon = 1

The current step was successful.

11: **inln** – **int32 scalar**

On intermediate exit: contains a flag indicating the action to be taken, if any, by the internal nonlinear equation solver.

12: **ires** – **int32 scalar**

On intermediate exit: with **irevcm** = 1, 2, 3, 4, 5, 6, 7 or 11, **ires** specifies the form of the residual to be returned by the RESID operation (see Section 3).

If **ires** = 1, then $r = g(t,y) - A(t,y)y'$ must be returned.

If **ires** = −1, then $r = -A(t,y)y'$ must be returned.

13: **irevcm** – **int32 scalar**

On intermediate exit: indicates what action you must take before re-entering d02nn. The possible exit values of **irevcm** are 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 or 11 which should be interpreted as follows:

irevcm = 1, 2, 3, 4, 5, 6, 7 or 11

Indicates that a RESID operation (see Section 3) is required: you must supply the residual of the system. For each of these values of **irevcm**, y_i is located in $\mathbf{y}(i)$, for $i = 1, 2, \ldots,$ **neq**.

For **irevcm** = 1, 3, 6 or 11, y'_i is located in $\mathbf{ydot}(i)$ and r_i should be stored in $\mathbf{rwork}(50 + 2 \times \mathbf{ldysav} + i)$, for $i = 1, 2, \ldots,$ **neq**.

For **irevcm** = 2, y'_i is located in $\mathbf{rwork}(50 + \mathbf{ldysav} + i)$ and r_i should be stored in $\mathbf{rwork}(50 + 2 \times \mathbf{ldysav} + i)$, for $i = 1, 2, \ldots,$ **neq**.

For **irevcm** = 4 or 7, y'_i is located in $\mathbf{ydot}(i)$ and r_i should be stored in $\mathbf{rwork}(50 + \mathbf{ldysav} + i)$, for $i = 1, 2, \ldots,$ **neq**.

For **irevcm** = 5, y'_i is located in $\mathbf{rwork}(50 + 2 \times \mathbf{ldysav} + i)$ and r_i should be stored in $\mathbf{ydot}(i)$, for $i = 1, 2, \ldots,$ **neq**.

irevcm = 8

Indicates that a JAC operation (see Section 3) is required: you must supply the Jacobian matrix.

If full matrix linear algebra is being used, then the (i,j)th element of the Jacobian must be stored in $\mathbf{wkjac}((j-1) \times \mathbf{neq} + i)$.

If banded matrix linear algebra is being used, then the (i,j)th element of the Jacobian must be stored in $\mathbf{wkjac}((i-1) \times m_B + k)$, where $m_B = m_L + m_U + 1$ and $k = \min(m_L - i + 1, 0) + j$; here m_L and m_U are the number of subdiagonals and superdiagonals, respectively, in the band.

If sparse matrix linear algebra is being used, then d02nr must be called to determine which column of the Jacobian is required and where it should be stored.

```
[j, iplace] = d02nr(inform);
```

will return in **j** the number of the column of the Jacobian that is required and will set **iplace** = 1 or 2 (see d02nr). If **iplace** = 1, then the (i,j)th element of the Jacobian must be stored in $\mathbf{rwork}(50 + 2 \times \mathbf{ldysav} + i)$; otherwise it must be stored in $\mathbf{rwork}(50 + \mathbf{ldysav} + i)$.

irevcm = 9

Indicates that a MONITR operation (see Section 3) can be performed.

irevcm = 10

Indicates that the current step was not successful, due to error test failure or convergence test failure. The only information supplied to you on this return is the current value of the variable t, located in $\mathbf{rwork}(19)$. No values must be changed before re-entering d02nn; this facility enables you to determine the number of unsuccessful steps.

On final exit: **irevcm** = 0 indicating that the user-specified task has been completed or an error has been encountered (see the descriptions for **itask** and **ifail**).

14: **lderiv(2) – logical array**

On final exit: $\mathbf{lderiv}(1)$ is normally unchanged. However if **itask** = 6 and internal initialization was successful then $\mathbf{lderiv}(1) =$ **true**.

$\mathbf{lderiv}(2) =$ **true**, if implicit equations were detected. Otherwise $\mathbf{lderiv}(2) =$ **false**.

15: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, the integrator detected an illegal input or that a linear algebra and/or integrator setup function has not been called prior to the call to the integrator. If **itrace** ≥ 0, the form of the error will be detailed on the current error message unit (see x04aa).

ifail $= 2$

The maximum number of steps specified has been taken (see the description of optional inputs in the integrator setup functions and the optional input continuation function, d02nz).

ifail $= 3$

With the given values of **rtol** and **atol** no further progress can be made across the integration range from the current point **t**. The components $\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(\mathbf{neq})$ contain the computed values of the solution at the current point **t**.

ifail $= 4$

There were repeated error test failures on an attempted step, before completing the requested task, but the integration was successful as far as **t**. The problem may have a singularity, or the local error requirements may be inappropriate.

ifail $= 5$

There were repeated convergence test failures on an attempted step, before completing the requested task, but the integration was successful as far as **t**. This may be caused by an inaccurate Jacobian matrix or one which is incorrectly computed.

ifail $= 6$

Some error weight w_i became zero during the integration (see the description of **itol**). Pure relative error control (**atol**$(i) = 0.0$) was requested on a variable (the ith) which has now vanished. The integration was successful as far as **t**.

ifail $= 7$

The user-supplied (sub)program **resid** operation (see Section 3) set the error flag **ires** $= 3$ continually despite repeated attempts by the integrator to avoid this.

ifail $= 8$

lderiv$(1) =$ **false** on entry but the internal initialization function was unable to initialize y' (more detailed information may be directed to the current error message unit, see x04aa).

ifail $= 9$

A singular Jacobian $\dfrac{\partial r}{\partial y}$ has been encountered. You should check the problem formulation and Jacobian calculation.

ifail $= 10$

An error occurred during Jacobian formulation or back-substitution (a more detailed error description may be directed to the current error message unit, see x04aa).

ifail $= 11$

The user-supplied (sub)program **resid** operation (see Section 3) signalled the integrator to halt the integration and return by setting **ires** $= 2$. Integration was successful as far as **t**.

ifail $= 12$

The (sub)program **monitr** operation (see Section 3) set **imon** $= -2$ and so forced a return but the integration was successful as far as **t**.

ifail $= 13$

The requested task has been completed, but it is estimated that a small change in **rtol** and **atol** is unlikely to produce any change in the computed solution. (Only applies when you are not operating in one step mode, that is when **itask** $\neq 2$ or 5.)

ifail $= 14$

The values of **rtol** and **atol** are so small that the function is unable to start the integration.

7 Accuracy

The accuracy of the numerical solution may be controlled by a careful choice of the parameters **rtol** and **atol**, and to a much lesser extent by the choice of norm. You are advised to use scalar error control unless the components of the solution are expected to be poorly scaled. For the type of decaying solution typical of many stiff problems, relative error control with a small absolute error threshold will be most appropriate (that is, you are advised to choose **itol** $= 1$ with **atol**(1) small but positive).

8 Further Comments

The cost of computing a solution depends critically on the size of the differential system and to a lesser extent on the degree of stiffness of the problem; also on the type of linear algebra being used. For further details see Section 8 of the documents for d02ng (full matrix), d02nh (banded matrix) or d02nj (sparse matrix).

In general, you are advised to choose the Backward Differentiation Formula option (setup function d02nv) but if efficiency is of great importance and especially if it is suspected that $\dfrac{\partial}{\partial y}\left(A^{-1}g\right)$ has complex eigenvalues near the imaginary axis for some part of the integration, you should try the BLEND option (setup function d02nw).

9 Example

```
neq = int32(3);
t = 0;
tout = 10;
y = [1; 0; 0];
ydot = [0; 0; 0];
rwork = zeros(62, 1);
rtol = [0.0001];
atol = [1e-07];
itol = int32(1);
inform = zeros(23, 1, 'int32');
ysave = zeros(3, 6);
wkjac = zeros(100,1);
jacpvt = zeros(150, 1, 'int32');
imon = int32(0);
inln = int32(0);
ires = int32(1);
irevcm = int32(0);
lderiv = [false; false];
itask = int32(3);
itrace = int32(0);
[const, rwork, ifail] = ...
    d02nv(int32(3), int32(6), int32(5), 'Newton', false, zeros(6), ...
    0, 1e-10, 10, 0, int32(200), int32(5), 'Average-L2', rwork);
[jacpvt, rwork, ifail] = ...
```

```
    d02nu(int32(3), int32(3), 'Analytical', int32(100), int32(0), ...
    int32(0), int32(150), 0, 0.1, 1e-4, true, rwork);
[tOut, toutOut, yOut, ydotOut, rworkOut, informOut, ysaveOut, wkjacOut,
...
 jacpvtOut, imonOut, inlnOut, iresOut, irevcmOut, lderivOut, ifail] = ...
    d02nn(neq, t, tout, y, ydot, rwork, rtol, atol, itol, inform, ...
      ysave, wkjac, jacpvt, imon, inln, ires, irevcm, lderiv, itask,
itrace)
```

```
tOut =
     0
toutOut =
    10
yOut =
     1
     0
     0
ydotOut =
     0
     0
     0
rworkOut =
     array elided
informOut =
           0
           0
           0
           0
           0
           0
           0
           0
           0
           0
           0
           0
           0
           0
           0
           0
           0
           0
           0
           0
           0
ysaveOut =
     1     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
wkjacOut =
     array elided
jacpvtOut =
     array elided
imonOut =
           0
inlnOut =
           0
iresOut =
          -1
irevcmOut =
          11
lderivOut =
     0
     0
ifail =
           1
```