NAG Toolbox

nag_ode_ivp_stiff_exp_revcom (d02nm)

1 Purpose

nag_ode_ivp_stiff_exp_revcom (d02nm) is a reverse communication function for integrating stiff systems of explicit ordinary differential equations.

2 Syntax

```
[t, y, ydot, rwork, inform, ysav, wkjac, jacpvt, imon, inln, ires, irevcm,
ifail] = nag_ode_ivp_stiff_exp_revcom(t, tout, y, ydot, rwork, rtol, atol, itol,
inform, ysav, wkjac, jacpvt, imon, inln, ires, irevcm, itask, itrace, 'neq',
neq, 'sdysav', sdysav, 'nwkjac', nwkjac)
[t, y, ydot, rwork, inform, ysav, wkjac, jacpvt, imon, inln, ires, irevcm,
ifail] = d02nm(t, tout, y, ydot, rwork, rtol, atol, itol, inform, ysav, wkjac,
jacpvt, imon, inln, ires, irevcm, itask, itrace, 'neq', neq, 'sdysav', sdysav,
'nwkjac', nwkjac)
```

Note: the interface to this routine has changed since earlier releases of the toolbox:

At Mark 22: *njcpvt* was removed from the interface; **neq** was made optional.

3 Description

nag_ode_ivp_stiff_exp_revcom (d02nm) is a general purpose function for integrating the initial value problem for a stiff system of explicit ordinary differential equations,

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

An outline of a typical calling program is given below:

```
%
 % data initialization
 %
 call linear algebra setup routine
 call integrator setup routine
 irevcm = int32(0);
 [..., irevcm, ...] = d02nm();
 while (irevcm > 0)
   if (irevcm == 8)
     supply the jacobian matrix                                    (i)
   elseif (irevcm == 9)
     perform monitoring tasks requested by the user               (ii)
   elseif (irecvm == 1 or irevcm >= 3 and irevcm <= 5)
     evaluate the derivative                                     (iii)
   elseif (irevcm == 10)
     indicates an unsuccessful step
   end
   [..., irevcm, ...] = d02nm();
 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 intermeditate return (**irevcm** $\neq 0$) from nag_ode_ivp_stiff_exp_revcom (d02nm); 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 argument **jceval** = 'A' (or **jceval** = 'F' if using sparse matrix linear algebra) in a call to the linear algebra setup function (see **jceval** in nag_ode_ivp_stiff_fulljac_setup (d02ns)). 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 argument 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 $\frac{d}{dy'}(\ ) = (hd)\frac{d}{dy}(\ )$.

The system of nonlinear equations that is solved has the form

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

but is solved in the form

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

where the function r is defined by

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

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

$$\frac{\partial r_i}{\partial y_j} = 1 - (hd)\frac{\partial g_i}{\partial y_j} \quad \text{if } i = j,$$

$$\frac{\partial r_i}{\partial y_j} = \quad - (hd)\frac{\partial g_i}{\partial y_j} \quad \text{otherwise,}$$

where t, h and d are located in **rwork**(19), **rwork**(16) and **rwork**(20) respectively and the array **y** contains the current values of the dependent variables. 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**, 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 timestep may be obtained by calling double function nag_ode_ivp_stiff_errest (d02za) as follows:

```
[errloc, ifail] = d02za(rwork(51+neq:51+neq+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 **monitr** in nag_ode_ivp_stiff_exp_fulljac (d02nb) for details on what optional input can be made.

If **y** is changed, then **imon** must be set to 2 before return to nag_ode_ivp_stiff_exp_revcom (d02nm). If either of the values of HMIN or HMAX are changed, then **imon** must be set ≥ 3 before return to nag_ode_ivp_stiff_exp_revcom (d02nm). If HNEXT is changed, then **imon** must be set to 4 before return to nag_ode_ivp_stiff_exp_revcom (d02nm).

In addition you can force nag_ode_ivp_stiff_exp_revcom (d02nm) to evaluate the residual vector

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

by setting **imon** $= 0$ and **inln** $= 3$ and then returning to nag_ode_ivp_stiff_exp_revcom (d02nm); on return to this monitoring operation the residual vector will be stored in **rwork**$(50 + 2 \times$ **neq** $+ i)$, for $i = 1, 2, \ldots,$ **neq**.

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

(iii) Evaluate the Derivative

This operation must evaluate the derivative vector for the explicit ordinary differential equation system defined by

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

where t is located in **rwork**(19).

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

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 argument **irevcm**. Between intermediate exits and re-entries, **all arguments other than ydot, rwork, wkjac, imon, inln and ires must remain unchanged**.

5.1 Compulsory Input Parameters

1: **t** – REAL (KIND=nag_wp)

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

2: **tout** – REAL (KIND=nag_wp)

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**.

3: **y**(**neq**) – REAL (KIND=nag_wp) array

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.

4: **ydot**(**neq**) – REAL (KIND=nag_wp) array

On intermediate re-entry: must be set to the derivatives as defined under the description of **irevcm**.

5: **rwork**$(\mathbf{50} + \mathbf{4} \times \mathbf{neq})$ – REAL (KIND=nag_wp) array

On initial entry: must be the same array as used by one of the method setup functions nag_ode_ivp_stiff_dassl (d02mv), nag_ode_ivp_stiff_bdf (d02nv) or nag_ode_ivp_stiff_blend (d02nw), and by one of the storage setup functions nag_ode_ivp_stiff_bandjac_setup (d02nt), nag_ode_ivp_stiff_sparjac_setup (d02nu) or nag_ode_ivp_stiff_bdf (d02nv). The contents of **rwork** must not be changed between any call to a setup function and the first call to nag_ode_ivp_stiff_exp_revcom (d02nm).

On intermediate re-entry: elements of **rwork** must be set to quantities as defined under the description of **irevcm**.

6: **rtol**(:) – REAL (KIND=nag_wp) array

The dimension of the array **rtol** must be at least 1 if **itol** $= 1$ or 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**).

7: **atol**(:) – REAL (KIND=nag_wp) array

The dimension of the array **atol** must be at least 1 if **itol** $= 1$ or 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**).

8: **itol** – INTEGER

On initial entry: a value to indicate the form of the local error test. **itol** indicates to nag_ode_ivp_stiff_exp_revcom (d02nm) 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**$(1) \times \lvert y_i \rvert +$ **atol**(1)
2	scalar	vector	**rtol**$(1) \times \lvert y_i \rvert +$ **atol**(i)
3	vector	scalar	**rtol**$(i) \times \lvert y_i \rvert +$ **atol**(1)
4	vector	vector	**rtol**$(i) \times \lvert y_i \rvert +$ **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: **itol** $= 1$, 2, 3 or 4.

9: **inform**$(\mathbf{23})$ – INTEGER array

10: **ysav**$(ldysav, \mathbf{sdysav})$ – REAL (KIND=nag_wp) array

$ldysav$, the first dimension of the array, must satisfy the constraint $ldysav \geq$ **neq**.

On initial entry: the second dimension of the array **ysav**. an appropriate value for **sdysav** is described in the specifications of the integrator setup functions nag_ode_ivp_stiff_bdf (d02nv) and nag_ode_ivp_stiff_blend (d02nw). This value must be the same as that supplied to the integrator setup function.

11: **wkjac**(**nwkjac**) – REAL (KIND=nag_wp) 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.

12: **jacpvt**($njcpvt$) – INTEGER array

On initial entry: the dimension of the array **jacpvt**. 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 nag_ode_ivp_stiff_bandjac_setup (d02nt) and nag_ode_ivp_stiff_sparjac_setup (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.

13: **imon** – INTEGER

On intermediate re-entry: may be reset to determine subsequent action in nag_ode_ivp_stiff_exp_revcom (d02nm).

imon $= -2$
Integration is to be halted. A return will be made from nag_ode_ivp_stiff_exp_revcom (d02nm) to the calling (sub)program with **ifail** $= 12$.

imon $= -1$
Allow nag_ode_ivp_stiff_exp_revcom (d02nm) 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 nag_ode_ivp_stiff_exp_revcom (d02nm) 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 solution **y**, provided by the **monitr** operation (see Section 3), will be used for the initial conditions.

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**.

14: **inln** – INTEGER

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 nag_ode_ivp_stiff_exp_revcom (d02nm), the residual vector is evaluated and placed in **rwork**$(50 + 2 \times$ **neq** $+ i)$, 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.

15: **ires** – INTEGER

On intermediate re-entry: should be unchanged unless one of the following actions is required of nag_ode_ivp_stiff_exp_revcom (d02nm) in which case **ires** should be set accordingly.

ires $= 2$
Indicates to nag_ode_ivp_stiff_exp_revcom (d02nm) that control should be passed back immediately to the calling (sub)program with the error indicator set to **ifail** $= 11$.

ires $= 3$
Indicates to nag_ode_ivp_stiff_exp_revcom (d02nm) 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 nag_ode_ivp_stiff_exp_revcom (d02nm) returns to the calling (sub)program with the error indicator set to **ifail** $= 7$.

ires $= 4$
Indicates to nag_ode_ivp_stiff_exp_revcom (d02nm) to stop its current operation and to enter the **monitr** operation (see Section 3) immediately.

16: **irevcm** – INTEGER

On initial entry: must contain 0.

On intermediate re-entry: should remain unchanged.

Constraint: **irevcm** $= 0$, 1, 3, 4, 5, 8, 9 or 10.

17: **itask** – INTEGER

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 before the first call to the integrator, or specified in the optional input function before a continuation call. **tcrit** (e.g., see nag_ode_ivp_stiff_bdf (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 nag_ode_ivp_stiff_bdf (d02nv)). **tcrit** must be specified under **itask** $= 4$.

Constraint: $1 \leq$ **itask** ≤ 5.

18: **itrace** – INTEGER

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 nag_file_set_unit_error (x04aa)).

itrace > 0
Warning messages are printed as above, and on the current advisory message unit (see nag_file_set_unit_advisory (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: **neq** – INTEGER

Default: the dimension of the arrays **y**, **ydot** and the first dimension of the array **ysav**. (An error is raised if these dimensions are not equal.)

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

Constraint: **neq** ≥ 1.

2: **sdysav** – INTEGER

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

On initial entry: the second dimension of the array **ysav**. an appropriate value for **sdysav** is described in the specifications of the integrator setup functions nag_ode_ivp_stiff_bdf (d02nv) and nag_ode_ivp_stiff_blend (d02nw). This value must be the same as that supplied to the integrator setup function.

3: **nwkjac** – INTEGER

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

On initial entry: the dimension of the array **wkjac**. 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 nag_ode_ivp_stiff_fulljac_setup (d02ns), nag_ode_ivp_stiff_bandjac_setup (d02nt) and nag_ode_ivp_stiff_sparjac_setup (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.

5.3 Output Parameters

1: **t** – REAL (KIND=nag_wp)

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

2: **y**(**neq**) – REAL (KIND=nag_wp) array

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

3: **ydot**(**neq**) – REAL (KIND=nag_wp) array

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

4: **rwork**($\mathbf{50} + \mathbf{4} \times$ **neq**) – REAL (KIND=nag_wp) array

On intermediate exit: contains information for JAC, FCN and MONITR operations as described in Section 3 and the argument **irevcm**.

5: **inform**(**23**) – INTEGER array

6: **ysav**($ldysav$, **sdysav**) – REAL (KIND=nag_wp) array

7: **wkjac**(**nwkjac**) – REAL (KIND=nag_wp) array

On intermediate exit: the Jacobian is overwritten.

8: **jacpvt**($njcpvt$) – INTEGER array

9: **imon** – INTEGER

On intermediate exit: used to pass information between nag_ode_ivp_stiff_exp_revcom (d02nm) and the MONITR operation (see Section 3). With **irevcm** $= 9$, **imon** contains a flag indicating under what circumstances the return from nag_ode_ivp_stiff_exp_revcom (d02nm) occurred:

imon $= -2$
Exit from nag_ode_ivp_stiff_exp_revcom (d02nm) after **ires** $= 4$ caused an early termination (this facility could be used to locate discontinuities).

imon $= -1$
The current step failed repeatedly.

imon $= 0$
Exit from nag_ode_ivp_stiff_exp_revcom (d02nm) after a call to the internal nonlinear equation solver.

imon $= 1$
The current step was successful.

10: **inln** – INTEGER

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

11: **ires** – INTEGER

On intermediate exit: with **irevcm** $= 1$, 2, 3, 4 or 5, **ires** contains the value 1.

12: **irevcm** – INTEGER

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

irevcm $= 1$, 3, 4 and 5
Indicates that an FCN operation (see Section 3) is required: $y' = g(t, y)$ must be supplied, where $\mathbf{y}(i)$ is located in y_i, for $i = 1, 2, \ldots,$ **neq**.

For **irevcm** $= 1$ or 3, y'_i should be placed in location **rwork**$(50 + 2 \times \mathbf{neq} + i)$, for $i = 1, 2, \ldots,$ **neq**.

For **irevcm** $= 4$, y'_i should be placed in location **rwork**$(50 + \mathbf{neq} + i)$, for $i = 1, 2, \ldots,$ **neq**.

For **irevcm** $= 5$, y'_i should be placed in location **ydot**(i), for $i = 1, 2, \ldots,$ **neq**.

irevcm $= 8$
Indicates that a JAC operation (see Section 3) is required: the Jacobian matrix must be supplied.

If full matrix linear algebra is being used, then the (i, j)th element of the Jacobian must be stored in **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 **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 super-diagonals, respectively, in the band.

If sparse matrix linear algebra is being used then nag_ode_ivp_stiff_sparjac_enq (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. If **iplace** $= 1$, then the (i, j)th element of the Jacobian must be stored in **rwork**$(50 + 2 \times \mathbf{neq} + i)$; otherwise it must be stored in **rwork**$(50 + \mathbf{neq} + i)$.

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 independent variable t, located in **rwork**(19). No values must be changed before re-entering nag_ode_ivp_stiff_exp_revcom (d02nm); this facility enables you to determine the number of unsuccessful steps.

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

13: **ifail** – INTEGER

On final exit: **ifail** $= 0$ unless the function detects an error (see Section 5).

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 nag_file_set_unit_error (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, nag_ode_ivp_stiff_contin (d02nz)).

ifail $= 3$ (*warning*)

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 $= 4$ (*warning*)

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 $= 5$ (*warning*)

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 $= 6$ (*warning*)

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

ifail $= 7$

The FCN operation (see Section 3) set the error flag **ires** $= 3$ continually despite repeated attempts by the integrator to avoid this.

ifail $= 8$

Not used for this integrator.

ifail $= 9$

A singular Jacobian $\dfrac{\partial r}{\partial y}$ has been encountered. This error exit is unlikely to be taken when solving explicit ordinary differential equations. 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 nag_file_set_unit_error (x04aa)).

ifail $= 11$ (*warning*)

The FCN 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 $= 12$ (*warning*)

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

ifail $= 13$ (*warning*)

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 nag_ode_ivp_stiff_exp_revcom (d02nm) is unable to start the integration.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

The accuracy of the numerical solution may be controlled by a careful choice of the arguments **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 9 of the documents for nag_ode_ivp_stiff_exp_fulljac (d02nb) (full matrix), nag_ode_ivp_stiff_exp_bandjac (d02nc) (banded matrix) or nag_ode_ivp_stiff_exp_sparjac (d02nd) (sparse matrix).

In general, you are advised to choose the backward differentiation formula option (setup function nag_ode_ivp_stiff_bdf (d02nv)) but if efficiency is of great importance and especially if it is suspected that $\dfrac{\partial g}{\partial y}$ has complex eigenvalues near the imaginary axis for some part of the integration, you should try the BLEND option (setup function nag_ode_ivp_stiff_blend (d02nw)).

9 Example

This example solves the well-known stiff Robertson problem

$$\begin{aligned} a' &= -0.04a + 1.0\text{e}4bc \\ b' &= 0.04a - 1.0\text{e}4bc - 3.0\text{e}7b^2 \\ c' &= 3.0\text{e}7b^2 \end{aligned}$$

over the range $[0, 10]$ with initial conditions $a = 1.0$ and $b = c = 0.0$ and with scalar error control (**itol** $= 1$). The integration proceeds until **tout** $= 10.0$ is passed, providing C^1 interpolation at intervals of 2.0 through a MONITR operation. The integration method used is the BDF method (setup function nag_ode_ivp_stiff_bdf (d02nv)) with a modified Newton method. The Jacobian is a full matrix, which is specified using the setup function nag_ode_ivp_stiff_fulljac_setup (d02ns); this Jacobian is to be calculated numerically.

9.1 Program Text

```
     function d02nm_example

fprintf('d02nm example results\n\n');

% Initialize setup setup variables and arrays.
neq    = nag_int(3);
neqmax = neq;
maxord = nag_int(5);
sdysav = maxord+1;
petzld = false;
tcrit  = 0;
hmin   = 1.0e-10;
hmax   = 10;
h0     = 0;
maxstp = nag_int(200);
mxhnil = nag_int(5);
const  = zeros(6, 1);
rwork  = zeros(50+4*neq, 1);

% BDF with Newton iterations.
[const, rwork, ifail] = d02nv( ...
                              neqmax, sdysav, maxord, 'Newton', petzld, ...
                              const, tcrit, hmin, hmax, h0, maxstp, ...
                              mxhnil, 'Average-L2', rwork);
% Numerical Jacobian
nwkjac = neqmax*(neqmax + 1);
[rwork, ifail] = d02ns( ...
                       neq, neqmax, 'Numerical', nwkjac, rwork);

% Initialize variables and arrays.
njcpvt = nag_int(1);
wkjac  = zeros(nwkjac, 1);
ydot   = zeros(neq, 1);
ysave  = zeros(neq, sdysav);
inform(1:23) = nag_int(0);
jacpvt(1) = nag_int(0);
algequ = zeros(neq, 1);

% Integrate to tout by overshooting (itask = 1).
% At monitoring stages output intermediate solutions after interpolating.
t     = 0;
tout  = 10.0;
itask = nag_int(1);
iout  = 1;
xout  = 2;
y     = [1; 0; 0];
itol  = nag_int(1);
rtol  = [1e-04];
atol  = [1e-07];

% Output header and initial results.
```

```
fprintf('\n      x           y(1)       y(2)          y(3)\n');
fprintf(' %8.3f %12.4f %12.2e %11.4f\n', t, y);

% Prepare to store results for plotting.
ncall = 1;
ykeep = y;
tkeep = [t];

% bogus initial entry values for unitialized input parameters.
irevcm = nag_int(-999); imon = irevcm; inln = irevcm; ires = irevcm;

% Ask for warning messages only.
itrace = nag_int(0);

% The reverse communication process is controlled by the value of irevcm
% (which is 0 for the final exit).
while (irevcm ~= 0)
  [t, y, ydot, rwork, inform, ysave, wkjac, jacpvt, imon, inln, ires, ...
   irevcm, ifail] = d02nm(...
                          t, tout, y, ydot, rwork, rtol, atol, itol, ...
                          inform, ysave, wkjac, jacpvt, imon, inln, ...
                          ires, irevcm, itask, itrace, 'neq', neq, ...
                          'sdysav', sdysav, 'nwkjac', nwkjac);

  % Evaluate the derivative, then use irevcm to place it.
  f(1) = -0.04*y(1) + 1.0e4*y(2)*y(3);
  f(2) =  0.04*y(1) - 1.0e4*y(2)*y(3) - 3.0e7*y(2)*y(2);
  f(3) =  3.0e7*y(2)*y(2);
  if (irevcm == 1 || irevcm == 3 )
    lrw = 50+2*neq;
    rwork(lrw+1:lrw+neq) = f;
  elseif (irevcm == 4)
    lrw = 50+neq;
    rwork(lrw+1:lrw+neq) = f;
  elseif (irevcm == 5)
    ydot = f;
  elseif (irevcm == 9 && imon == 1)
    % Extract useful information about the current step.
    tc    = rwork(19);
    hlast = rwork(15);
    hnext = rwork(16);
    nqu   = nag_int(rwork(10));

    % For low values of the independent variable, store the results
    % for plotting.
    if tc < 2.0
      ncall = ncall + 1;
      tkeep(ncall,1) = tc;
      ykeep(:,ncall) = y;
    end

    % xout is intermediate output point.
    while (xout <= tc)
      lrw = 50+neq;
      [sol, ifail] = d02xk(...
                           xout, neq, ysave, rwork(lrw+1:lrw+neq), ...
                           tc, nqu, hlast, hnext);
      % Output interpolated result, and save it for plotting.
      fprintf(' %8.3f %12.4f %12.2e %11.4f\n', xout, sol);
      ncall = ncall + 1;
      tkeep(ncall) = xout;
      ykeep(:,ncall) = sol;
      iout = iout + 1;
      xout = iout*2;
    end
  end
end

% d02ny is an integrator diagnostic routine which can be called
% after d02nm.  Output results.
[hu, h, tcur, tolsf, nst, nre, nje, nqu, nq, niter, imxer, algequ, ifail] ...
```

```
  = d02ny(...
          neq, neqmax, rwork, inform);
fprintf('\nDiagnostic information\n integration status:\n');
fprintf('   last and next step sizes = %8.5f, %8.5f\n',hu, h);
fprintf('   integration stopped at x = %8.5f\n',tcur);
fprintf(' algorithm statistics:\n');
fprintf('   number of time-steps and Newton iterations  = %5d %5d\n', ...
        nst,niter);
fprintf('   number of residual and jacobian evaluations = %5d %5d\n', ...
        nre,nje);
fprintf('   order of method last used and next to use   = %5d %5d\n', ...
        nqu,nq);
fprintf('   component with largest error                = %5d\n',imxer);
% Plot results.
fig1 = figure;
display_plot(tkeep, ykeep)

function display_plot(x, y)
  % Plot one of the curves and then add the other two.
  hline1 = semilogx(x, y(1,:));
  hold on
  [haxes, hline2, hline3] = plotyy(x,y(3,:),x,y(2,:), ...
                                   'semilogx','semilogx');
  % Set the axis limits and the tick specifications to beautify the plot.
  set(haxes(1), 'YLim', [-0.05 1.3]);
  set(haxes(1), 'XMinorTick', 'on', 'YMinorTick', 'on');
  set(haxes(1), 'YTick', [0.0:0.2:1.2]);
  set(haxes(2), 'YLim', [0.0 5e-5]);
  set(haxes(2), 'YMinorTick', 'on');
  set(haxes(2), 'YTick', [5e-6:5e-6:3.5e-5]);
  for iaxis = 1:2
    % These properties must be the same for both sets of axes.
    set(haxes(iaxis), 'XLim', [0 12]);
    set(haxes(iaxis), 'XTick', [0.0001 0.001 0.01 0.1 1 10]);
  end
  set(gca, 'box', 'off');
  % Add title.
  title({'Stiff Robertson Problem (Explicit ODE):', ...
         'BDF with Newton Iterations'},'Position',[0.05,1.1]);
  % Label the x axis, and both y axes.
  xlabel('x');
  ylabel(haxes(1),'Solution (a,c)');
  ylabel(haxes(2),'Solution (b)');
  % Add a legend.
  legend({'a','c','b'},'Position',[0.45,0.45,0.15,0.15]);
  % Set some features of the three lines.
  set(hline1, 'Marker', '+','Linestyle','-','Color','red');
  set(hline2, 'Marker', 'o','Linestyle',':','Color','green');
  set(hline3, 'Marker', 'x','Linestyle','--','Color','blue');
  hold off
```

9.2 Program Results

```
     d02nm example results

      x           y(1)       y(2)          y(3)
    0.000       1.0000     0.00e+00      0.0000
    2.000       0.9416     2.70e-05      0.0584
    4.000       0.9055     2.24e-05      0.0945
    6.000       0.8793     1.96e-05      0.1207
    8.000       0.8585     1.77e-05      0.1414
   10.000       0.8414     1.62e-05      0.1586

Diagnostic information
 integration status:
   last and next step sizes =  0.90178,  0.90178
   integration stopped at x = 10.76621
```

```
algorithm statistics:
  number of time-steps and Newton iterations  =    55    78
  number of residual and jacobian evaluations =   128    16
  order of method last used and next to use   =     4     4
  component with largest error                =     3
```