NAG Toolbox for MATLAB

d02lx

1 Purpose

d02lx is a setup function which must be called by you prior to the first call of the integrator d02la and may be called prior to any further call to d02la.

2 Syntax

```
[start, rwork, ifail] = d02lx(h, tol, thres, thresp, maxstp, start, onestp, high, rwork, 'neq', neq, 'lrwork', lrwork)
```

3 Description

d02lx permits you to set optional inputs prior to any call of d02la. It must be called before the first call of function d02la and it may be called before any continuation call of function d02la.

4 References

None.

5 Parameters

5.1 Compulsory Input Parameters

1: **h – double scalar**

If **start** = **true**, **h** may specify an initial step size to be attempted in d02la.

If **start** = **false**, then **h** may specify a step size to override the choice of next step attempted made internally to d02la.

The sign of **h** is not important, as the absolute value of **h** is chosen and the appropriate sign is selected by d02la.

If this option is not required then you must set **h** = 0.0.

2: **tol – double scalar**

Must be set to a relative tolerance for controlling the error in the integration by d02la. d02la has been designed so that, for most problems, a reduction in **tol** leads to an approximately proportional reduction in the error in the solution. However the actual relation between **tol** and the accuracy of the solution cannot be guaranteed. You are strongly recommended to repeat the integration with a smaller value of **tol** and compare the results. See the description of **thres** and **thresp** for further details of how **tol** is used.

Constraint: $10 \times \epsilon \leq$ **tol** ≤ 1.0 (ϵ is the ***machine precision***, see x02aj).

3: **thres**(**neq**) **– double array**

4: **thresp**(**neq**) **– double array**

thres and **thresp** may be set to thresholds for use in the error control of d02la. At each step in the numerical integration estimates of the local errors E1(i) and E2(i) in the solution, y_i, and its derivative, y_i', respectively are computed, for $i = 1, 2, \ldots,$ **neq**. For the step to be accepted conditions of the following type must be satisfied:

$$\max_{1\leq i\leq \mathbf{neq}} \left(\frac{\mathrm{E1}(i)}{\max(\mathbf{thres}(i), |y_i|)} \right) \leq \mathbf{tol},$$

$$\max_{1\leq i\leq \mathbf{neq}} \left(\frac{\mathrm{E2}(i)}{\max(\mathbf{thresp}(i), |y'_i|)} \right) \leq \mathbf{tol}.$$

If one or both of these is not satisfied then the step size is reduced and the solution is recomputed.If $\mathbf{thres}(1) \leq 0.0$ on entry, then a value of $50.0 \times \epsilon$ is used for $\mathbf{thres}(i)$, for $i = 1, 2, \ldots, \mathbf{neq}$, where ϵ is ***machine precision***. Similarly for **thresp**.

Constraints:

$\mathbf{thres}(1) \leq 0.0$ or $\mathbf{thres}(i) > 0.0$, for $i = 1, 2, \ldots, \mathbf{neq}$;
$\mathbf{thresp}(1) \leq 0.0$ or $\mathbf{thresp}(i) > 0.0$, for $i = 1, 2, \ldots, \mathbf{neq}$.

5: **maxstp – int32 scalar**

A bound on the number of steps attempted in any one call of d02la.

If $\mathbf{maxstp} \leq 0$ on entry, a value of 1000 is used.

6: **start – logical scalar**

Specifies whether or not the call of d02la is for a new problem. **start** = **true** indicates that a new problem is to be solved. **start** = **false** indicates the call of d02lx is prior to a continuation call of d02la.

7: **onestp – logical scalar**

The mode of operation for d02la.

onestp = **true**

d02la will operate in one-step mode, that is it will return after each successful step.

onestp = **false**

d02la will operate in interval mode, that is it will return at the end of the integration interval.

8: **high – logical scalar**

If **high** = **true**, a high-order method will be used, whereas if **high** = **false**, a low-order method will be used. (See the specification of d02la for further details.)

9: **rwork**(**lrwork**) **– double array**

This **must** be the same parameter **rwork** supplied to d02la. It is used to pass information to d02la and therefore the contents of this array **must not** be changed before calling d02la.

5.2 Optional Input Parameters

1: **neq – int32 scalar**

Default: The dimension of the arrays **thres**, **thresp**. (An error is raised if these dimensions are not equal.)

the number of second-order ordinary differential equations to be solved by d02la.

Constraint: $\mathbf{neq} \geq 1$.

2: **lrwork – int32 scalar**

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

Constraints:

if **high** = **true**, **lrwork** $\geq 16 + 20 \times$ **neq**;
if **high** = **false**, **lrwork** $\geq 16 + 11 \times$ **neq**.

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **start – logical scalar**

start = **false**.

2: **rwork**(**lrwork**) **– double array**

This **must** be the same parameter **rwork** supplied to d02la. It is used to pass information to d02la and therefore the contents of this array **must not** be changed before calling d02la.

3: **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

thres$(1) > 0.0$ and for some i **thres**$(i) \leq 0.0$, $1 \leq i \leq$ **neq**, and/or, **thresp**$(1) > 0.0$ and for some i **thresp**$(i) \leq 0.0$, $1 \leq i \leq$ **neq**.

ifail = 2

lrwork is too small.

ifail = 3

tol does not satisfy $10 \times \epsilon \leq$ **tol** ≤ 1.0 (ϵ is the ***machine precision***, see x02aj)

7 Accuracy

Not applicable.

8 Further Comments

Prior to a continuation call of d02la, you may reset **any** of the optional parameters by calling d02lx with **start** = **false**.

You may reset:

h	to override the internal step size selection;
tol, **thres**, **thresp**	to change the error requirements;
maxstp	to increase or decrease the number of steps attempted before an error exit is returned;
onestp	to change the mode of operation of d02la;
high	to change the order of the method being used.

9 Example

d02la_fcn.m

```
function f = fcn(neq, t, y)
  r = sqrt(y(1)^2+y(2)^2)^3;
  f = zeros(2,1);
  f(1) = -y(1)/r;
  f(2) = -y(2)/r;
```

```
t = 0;
tend = 20;
y = [0.5;
     0];
yp = [0;
     1.732050807568877];
ydp = zeros(2, 1);
rwork = zeros(56,1);
[startOut, rwork, ifail] = ...
     d02lx(0, 1e-4, zeros(2,1), zeros(2,1), int32(0), true, true, false,
rwork);
[tOut, yOut, ypOut, ydpOut, rworkOut, ifail] = ...
    d02la('d02la_fcn', t, tend, y, yp, ydp, rwork)
```

```
tOut =
    0.0624
yOut =
    0.4923
    0.1075
ypOut =
   -0.2464
    1.7054
ydpOut =
   -3.8480
   -0.8406
rworkOut =
     array elided
ifail =
           0
```