# NAG Library Function Document

# nag_ode_ivp_adams_setup (d02qwc)

## 1 Purpose

nag_ode_ivp_adams_setup (d02qwc) is a setup function which must be called prior to the first call of the integration function nag_ode_ivp_adams_roots (d02qfc) and may be called prior to any subsequent continuation call of the integrator.

## 2 Specification

```
#include <nag.h>
#include <nagd02.h>
void nag_ode_ivp_adams_setup (Nag_Start *state, Integer neqf,
     Nag_Boolean vectol, const double atol[], const double rtol[],
     Nag_Boolean one_step, Nag_Boolean crit, double tcrit, double hmax,
     Integer max_step, Integer neqg, Nag_Boolean *alter_g,
     Nag_Boolean sophist, Nag_ODE_Adams *opt, NagError *fail)
```

## 3 Description

nag_ode_ivp_adams_setup (d02qwc) permits initialization of the integration method and setting of integration inputs prior to any call of nag_ode_ivp_adams_roots (d02qfc).

It must be called before the first call of the function nag_ode_ivp_adams_roots (d02qfc) and it may be called before any continuation call of nag_ode_ivp_adams_roots (d02qfc).

## 4 References

None.

## 5 Arguments

1: **state** – Nag_Start * *Input/Output*

*On entry*: specifies whether the integration function nag_ode_ivp_adams_roots (d02qfc) is to start a new system of ordinary differential equations, restart a system or continue with a system.

**state** = Nag_NewStart
Start integration with a new differential system.

**state** = Nag_ReStart
Restart integration with the current differential system.

**state** = Nag_Continue
Continue integration with the current differential system.

*Constraint*: **state** = Nag_NewStart, Nag_ReStart or Nag_Continue.

*On exit*: **state** is set to Nag_Continue, except that if an error is detected, **state** is unchanged.

2: **neqf** – Integer *Input*

*On entry*: the number of ordinary differential equations to be solved by the integration function. **neqf** must remain unchanged on subsequent calls to nag_ode_ivp_adams_setup (d02qwc) with **state** = Nag_Continue or Nag_ReStart.

*Constraint*: **neqf** $\geq 1$.

3: **vectol** – Nag_Boolean *Input*

*On entry*: specifies whether vector or scalar error control is to be employed for the local error test in the integration. If **vectol** = Nag_TRUE, then vector error control will be used and you must specify values of **rtol**$[i]$ and **atol**$[i]$, for $i = 0, 1, \ldots,$ **neqf** $- 1$. Otherwise scalar error control will be used and you must specify values of just **rtol**$[0]$ and **atol**$[0]$.

The error test to be satisfied is of the form

$$\sqrt{\sum_{i=1}^{\mathbf{neqf}} \left(\frac{e_i}{w_i}\right)^2} \leq 1.0,$$

where $w_i$ is defined as follows:

| **vectol** | $w_i$ |
|---|---|
| Nag_TRUE | **rtol**$[i-1] \times \lvert y_i \rvert +$ **atol**$[i-1]$ |
| Nag_FALSE | **rtol**$[0] \times \lvert y_i \rvert +$ **atol**$[0]$ |

and $e_i$ is an estimate of the local error in $y_i$, computed internally. **vectol** must remain unchanged on subsequent calls to nag_ode_ivp_adams_setup (d02qwc) with **state** = Nag_Continue or Nag_ReStart.

4: **atol**[**neqf**] – const double *Input*

*On entry*: the absolute local error tolerance (see **vectol**).

*Constraint*: **atol**$[i] \geq 0.0$.

5: **rtol**[**neqf**] – const double *Input*

*On entry*: the relative local error tolerance (see **vectol**).

*Constraints*:

**rtol**$[i] \geq 0.0$;
if **atol**$[i] = 0.0$, **rtol**$[i] \geq 4.0 \times$ ***machine precision***.

6: **one_step** – Nag_Boolean *Input*

*On entry*: the mode of operation of the integration function. If **one_step** = Nag_TRUE, the integration function will operate in one-step mode, that is it will return after each successful step. Otherwise the integration function will operate in interval mode, that is it will return at the end of the integration interval.

7: **crit** – Nag_Boolean *Input*

*On entry*: specifies whether or not there is a value for the independent variable beyond which integration is not to be attempted. Setting **crit** = Nag_TRUE indicates that there is such a point, whereas **crit** = Nag_FALSE indicates that there is no such restriction.

8: **tcrit** – double *Input*

*On entry*: with **crit** = Nag_TRUE, **tcrit** must be set to a value of the independent variable beyond which integration is not to be attempted. Otherwise **tcrit** is not referenced.

9: **hmax** – double *Input*

*On entry*:

**hmax** $\neq 0.0$
A bound on the absolute step size during the integration is taken to be $|$**hmax**$|$.

**hmax** $= 0.0$
No bound is assumed on the step size during the integration.

A bound may be required if there are features of the solution on very short ranges of integration which may be missed. You should try **hmax** = 0.0 first.

Note: this argument only affects the step size if the option **crit** = Nag_TRUE is being used.

10: **max_step** – Integer *Input*

*On entry*: a bound on the number of attempted steps in any one call to the integration function.

If **max_step** $\leq 0$ on entry, a value of 1000 is used.

11: **neqg** – Integer *Input*

*On entry*: specifies whether or not root-finding is required in nag_ode_ivp_adams_roots (d02qfc).

**neqg** $\leq 0$
No root-finding is attempted.

**neqg** $> 0$
Root-finding is required and **neqg** event functions will be specified for the integration function.

12: **alter_g** – Nag_Boolean * *Input/Output*

*On entry*: specifies whether or not the event functions have been redefined. **alter_g** need not be set if **state** = Nag_NewStart. On subsequent calls to nag_ode_ivp_adams_setup (d02qwc), if **neqg** has been set positive, then **alter_g** = Nag_FALSE specifies that the event functions remain unchanged, whereas **alter_g** = Nag_TRUE specifies that the event functions have changed. Because of the expense in reinitializing the root searching procedure, **alter_g** should be set to Nag_TRUE only if the event functions really have been altered. **alter_g** need not be set if the root-finding option is not used.

*On exit*: **alter_g** is set to Nag_FALSE, except that if an error is detected, **alter_g** is unchanged.

13: **sophist** – Nag_Boolean *Input*

*On entry*: the type of search technique to be used in the root-finding.

**sophist** = Nag_TRUE
A sophisticated and reliable but expensive technique will be used, whereas for **sophist** = Nag_FALSE a simple but less reliable technique will be used.

**neqg** $\leq 0$
**sophist** is not referenced.

14: **opt** – Nag_ODE_Adams * *Output*

*On exit*: the structure of type Nag_ODE_Adams will have been initialized to appropriate values for entry to the integration function nag_ode_ivp_adams_roots (d02qfc). **opt** must be passed unchanged to the integration function.

Memory will have been allocated by nag_ode_ivp_adams_setup (d02qwc) to several pointers within **opt**, this memory is used by the integration function nag_ode_ivp_adams_roots (d02qfc) and the interpolation function nag_ode_ivp_adams_interp (d02qzc). The library function nag_ode_ivp_adams_free (d02qyc) is provided so that this memory can be freed when all calls to nag_ode_ivp_adams_roots (d02qfc) and nag_ode_ivp_adams_interp (d02qzc) have been completed. A call to nag_ode_ivp_adams_free (d02qyc) may also be made prior to reentering nag_ode_ivp_adams_setup (d02qwc) with **state** = Nag_NewStart.

15: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.

**NE_BAD_PARAM**

On entry, argument **state** had an illegal value.

**NE_BOOL_NOT_SET**

The Boolean argument **crit** has not been set to Nag_TRUE or Nag_FALSE.

**NE_INT_ARG_LT**

On entry, **neqf** $= \langle value \rangle$.
Constraint: **neqf** $\geq 1$.

**NE_NEQF_CHANGED**

**state** $= \langle string \rangle$ but **neqf** has been changed. **neqf** was $\langle value \rangle$ but is now $\langle value \rangle$.

**NE_NEQG_CHANGED**

**alter_g** = Nag_FALSE but **neqg** has been changed. **neqg** was $\langle value \rangle$ but is now $\langle value \rangle$.

**NE_REAL_ARG_LT**

On entry, **atol**$[\langle value \rangle]$ must not be less than 0.0: **atol**$[\langle value \rangle] = \langle value \rangle$.

On entry, **rtol**$[\langle value \rangle]$ must not be less than 0.0: **rtol**$[\langle value \rangle] = \langle value \rangle$.

**NE_REAL_LT_COND**

When **atol**$[\langle value \rangle] = 0.0$, **rtol**$[\langle value \rangle]$ must not be less than $4 \times \epsilon$. **rtol**$[\langle value \rangle] = \langle value \rangle$, $4 \times \epsilon = \langle value \rangle$.

**NE_STATE**

**state** $\neq$ Nag_NewStart on first call.

**NE_VECTOL_CHANGED**

**state** $= \langle string \rangle$ but **vectol** has been changed. **vectol** was $\langle string \rangle$ but is now $\langle string \rangle$.

## 7 Accuracy

Not applicable.

## 8 Parallelism and Performance

nag_ode_ivp_adams_setup (d02qwc) is not threaded in any implementation.

## 9 Further Comments

Prior to a continuation call of the integration function, you may reset some of the arguments by calling nag_ode_ivp_adams_setup (d02qwc) with **state** = Nag_Continue. You may reset:

(a) **hmax** - to alter the maximum step size selection;

(b) **rtol**, **atol** - to change the error requirements;

(c) **max_step** - to increase or decrease the number of attempted steps before an error exit is returned;

(d) **one_step** - to change the operation mode of the integration function;

(e) **crit**, **tcrit** - to alter the point beyond which integration must not be attempted; and

(f) **neqg**, **alter_g**, **sophist**
- to alter the number and type of event functions, and also the search method.

If the behaviour of the system of differential equations has altered and you wish to restart the integration method from the value of **t** output from the integration function, then set **state** = Nag_ReStart and some of the integration arguments may be reset also. If you want to redefine the system of differential equations or start a new integration problem, then set **state** = Nag_NewStart. Resetting **state** = Nag_ReStart or Nag_NewStart on normal continuation calls causes a restart in the integration process, which is very inefficient when not needed.

## 10 Example

See example program for nag_ode_ivp_adams_roots (d02qfc).