NAG Toolbox

nag_quad_opt_set (d01zk)

1 Purpose

nag_quad_opt_set (d01zk) either initializes or resets the optional parameter arrays or sets a single optional parameter for supported problem solving functions in Chapter D01.

2 Syntax

```
[iopts, opts, ifail] = nag_quad_opt_set(optstr, iopts, opts, 'liopts', liopts, 'lopts', lopts)
[iopts, opts, ifail] = d01zk(optstr, iopts, opts, 'liopts', liopts, 'lopts', lopts)
```

3 Description

nag_quad_opt_set (d01zk) has three purposes: to initialize optional parameter arrays; to reset all optional parameters to their default values; or to set a single optional parameter to a user-supplied value.

Optional parameters and their values are, in general, presented as a character string, **optstr**, of the form '*option* = *optval*'; alphabetic characters can be supplied in either upper or lower case. Both *option* and *optval* may consist of one or more tokens separated by white space. The tokens that comprise *optval* will normally be either an integer, real or character value as defined in the description of the specific optional argument. In addition all optional parameters can take an *optval* DEFAULT which resets the optional parameter to its default value.

It is imperative that optional parameter arrays are initialized before any options are set, before the relevant problem solving function is called and before any options are queried using nag_quad_opt_get (d01zl). To initialize the optional parameter arrays **iopts** and **opts** for a specific problem solving function, the option **Initialize** is used with *optval* identifying the problem solving function to be called, via its short name. For example, to initialize the optional parameter arrays to be passed to nag_quad_1d_gen_vec_multi_rcomm (d01ra) and its associated function nag_quad_1d_gen_vec_multi_dimreq (d01rc), nag_quad_opt_set (d01zk) is called as follows:

```
[iopts, opts, ifail] = d01zk('Initialize = d01ra', iopts, opts);
```

The available option names and their corresponding valid values are given in Section 11 in nag_quad_md_sgq_multi_vec (d01es) and nag_quad_1d_gen_vec_multi_rcomm (d01ra).

4 References

None.

5 Parameters

5.1 Compulsory Input Parameters

1: **optstr** – CHARACTER(*)

A string identifying the option to be set.

Initialize = *function name*
Initialize the optional parameter arrays **iopts** and **opts** for use with function *function name*, where *function name* is the short name associated with the function of interest.

Defaults
Resets all options to their default values.

option = *optval*
See Section 11 in nag_quad_md_sgq_multi_vec (d01es) and nag_quad_1d_gen_vec_multi_rcomm (d01ra) for details of valid values for *option* and *optval*. The equals sign (=) delimiter must be used to separate the *option* from its *optval* value.

optstr is case insensitive. Each token in the *option* and *optval* component must be separated by at least one space.

2: **iopts**(**liopts**) – INTEGER array

Optional parameter array.

If **optstr** has the form **Initialize** = *function name*, the contents of **iopts** need not be set.

Otherwise, **iopts must not** have been altered since the last call to nag_quad_opt_set (d01zk), nag_quad_opt_get (d01zl) or the selected problem solving function.

3: **opts**(**lopts**) – REAL (KIND=nag_wp) array

Optional parameter array.

If **optstr** has the form **Initialize** = *function name*, the contents of **opts** need not be set.

Otherwise, **opts must not** have been altered since the last call to nag_quad_opt_set (d01zk), nag_quad_opt_get (d01zl) or the selected problem solving function.

5.2 Optional Input Parameters

1: **liopts** – INTEGER

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

The length of the array **iopts**.

Constraint: unless otherwise stated in the documentation for a specific, supported, problem solving function, **liopts** ≥ 100.

2: **lopts** – INTEGER

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

The length of the array **opts**.

Constraint: unless otherwise stated in the documentation for a specific, supported, problem solving function, **lopts** ≥ 100.

5.3 Output Parameters

1: **iopts**(**liopts**) – INTEGER array

Dependent on the contents of **optstr**, either an initialized, reset or updated version of the optional parameter array.

2: **opts**(**lopts**) – REAL (KIND=nag_wp) array

Dependent on the contents of **optstr**, either an initialized, reset or updated version of the optional parameter array.

3: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 11$ (*warning*)

On entry, the optional parameter in **optstr** was not recognized.

ifail $= 12$ (*warning*)

On entry, the expected delimiter '=' was not found in **optstr**.

ifail $= 13$ (*warning*)

On entry, could not convert the specified *optval* to an integer.

On entry, could not convert the specified *optval* to a real.

ifail $= 14$

On entry, attempting to initialize the optional parameter arrays but specified function name was not valid.

ifail $= 15$ (*warning*)

On entry, the *optval* supplied for the integer optional parameter is not valid.

ifail $= 16$ (*warning*)

On entry, the *optval* supplied for the real optional parameter is not valid.

ifail $= 17$ (*warning*)

On entry, the *optval* supplied for the character optional parameter is not valid.

ifail $= 21$

On entry, either the option arrays have not been initialized or they have been corrupted.

ifail $= 31$

liopts is too small.

ifail $= 51$

lopts is too small.

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

Not applicable.

8 Further Comments

For suites of functions that share the same option arrays, the option arrays must be initialized using the primary (driver) function name. For example for functions nag_quad_1d_gen_vec_multi_rcomm (d01ra) and nag_quad_1d_gen_vec_multi_dimreq (d01rc), the option arrays must be initialized for nag_quad_1d_gen_vec_multi_rcomm (d01ra).

When encoding integer valued options in **optstr**, the integer *optval* must be written as an explicit integer. For example, `"Maximum Subdivisions = 12"` is acceptable, whereas `"Maximum Subdivisions = 12.0"` and `"Maximum Subdivisions = 0.12E2"` are not.

When encoding real valued options in **optstr**, the *optval* may be integral if appropriate. For example, `"Absolute Tolerance = 10"`, `"Absolute Tolerance = 10.0"` and `"Absolute Tolerance = 1.0E1"` are all acceptable.

9 Example

See the example programs associated with the problem solving function you wish to use for a demonstration of how to use nag_quad_opt_set (d01zk) to initialize option arrays and set options.

9.1 Program Text

```
function d01zk_example

fprintf('d01zk example results\n\n');

  % Setup phase.

  % set problem parameters
  ni = nag_int(2);
  nx = nag_int(0);
  % lower (a) and upper (b) bounds
  a = 0;
  b = pi;
  iopts = zeros(100, 1, nag_int_name);
  opts  = zeros(100, 1);

  % initialize option arrays
  [iopts, opts, ifail] = d01zk('Initialize = d01ra', iopts, opts);

  % set any non-default options required
  [iopts, opts, ifail] = d01zk('Quadrature Rule = gk41', iopts, opts);
  [iopts, opts, ifail] = d01zk('Absolute Tolerance = 1.0e-7', iopts, opts);
  [iopts, opts, ifail] = d01zk('Relative Tolerance = 1.0e-7', iopts, opts);

  % determine maximum required array lengths
  [lenxrq, ldfmrq, sdfmrq, licmin, licmax, lcmin, lcmax, ifail] = ...
        d01rc(ni, iopts, opts);

  % allocate remaining arrays
  needi  = zeros(ni, 1, nag_int_name);
  comm   = zeros(lcmax, 1);
  icomm  = zeros(licmax, 1, nag_int_name);
  fm     = zeros(ldfmrq, sdfmrq);
  dinest = zeros(ni, 1);
  errest = zeros(ni, 1);
  x      = zeros(1, lenxrq);
```

```
% Solve phase.

% Use d01ra to evaluate the definate integrals of:
%   f_1 = (x*sin(2*x))*cos(15*x)
%   f_2 = (x*sin(2*x))*(x*cos(50*x))

% set initial irevcm
irevcm = nag_int(1);

while irevcm ~= 0
  [irevcm, sid, needi, x, nx, dinest, errest, icomm, comm, ifail] = ...
    d01ra(irevcm, a, b, needi, x, nx, fm, dinest, errest, ...
          iopts, opts, icomm, comm);

  switch irevcm
    case 11
      % Initial returns.
      % These will occur during the non-adaptive phase.
      % All values must be supplied.
      % dinest and errest do not contain approximations
      % over the complete interval at this stage.

      % Calculate x*sin(2*x), storing the result in fm(2,1:nx) for re-use.
      fm(2, :) = x.*sin(2*x);

      % Calculate f_1
      fm(1, :) = fm(2, :).*cos(15*x);

      % Calculate f_2
      fm(2, :) = fm(2, :).*x.*cos(50*x);
    case 12
      % Intermediate returns.
      % These will occur during the adaptive phase.
      % All requested values must be supplied.
      % dinest and errest do not contain approximations
      % over the complete interval at this stage.

      % Calculate x*sin(2*x).
      fm(2, :) = x.*sin(2*x);

      % Calculate f_1 if required
      if needi(1) == 1
        fm(1, :) = fm(2, :).*cos(15*x);
      end

      % Complete f_2 calculation if required.
      if needi(2) == 1
        fm(2, :) = fm(2, :).*x.*cos(50*x);
      end
    case 0
      % Final return
  end
end

% query some currently set options and statistics.
[ivalue, rvalue, cvalue, optype, ifail] = ...
   d01zl('Quadrature rule', iopts, opts);
display_option('Quadrature rule',optype,ivalue,rvalue,cvalue);
[ivalue, rvalue, cvalue, optype, ifail] = ...
   d01zl('Maximum Subdivisions', iopts, opts);
display_option('Maximum Subdivisions',optype,ivalue,rvalue,cvalue);
[ivalue, rvalue, cvalue, optype, ifail] = ...
   d01zl('Extrapolation', iopts, opts);
display_option('Extrapolation',optype,ivalue,rvalue,cvalue);
[ivalue, rvalue, cvalue, optype, ifail] = ...
   d01zl('Extrapolation Safeguard', iopts, opts);
display_option('Extrapolation Safeguard',optype,ivalue,rvalue,cvalue);

% print solution
fprintf('\nIntegral |  needi  |   dinest   |   errest   \n');
```

```
  for j=1:ni
    fprintf('%9d %9d %12.4e %12.4e\n', j, needi(j), dinest(j), errest(j));
  end

function [dinest, errest, user] = monit(ni, ns, dinest, errest, fcount, ...
                                        sinfoi, evals, ldi, sinfor, fs, ...
                                        es, ldr, user)
  % Display information on individual segments
  fprintf('\nInformation on splitting and evaluations over subregions.\n');
  for k=1:ns
    sid = sinfoi(1,k);
    parent = sinfoi(2,k);
    child1 = sinfoi(3,k);
    child2 = sinfoi(4,k);
    level = sinfoi(5,k);
    lbnd = sinfor(1,k);
    ubnd = sinfor(2,k);
    fprintf('\nSegment %3d Sid = %3d', k, sid);
    fprintf(' Parent = %3d Level = %3d.\n', parent, level);
    if (child1>0)
      fprintf('Children = (%3d, %3d)\n', child1, child2);
    end
    fprintf('Bounds (%11.4e, %11.4e)\n', lbnd, ubnd);
    for j = 1:ni
      if (evals(j,k) ~= 0)
        fprintf('Integral %2d approximation %11.4e\n', j, fs(j,k));
        fprintf('Integral %2d error estimate %11.4e\n', j, es(j,k));
      end
      if (evals(j,k) ~= 1)
        fprintf('Integral %2d evaluation', j);
        fprintf(' has been superseded by descendants.\n');
      end
    end
  end

function display_option(optstr,optype,ivalue,rvalue,cvalue)
  % Query optype and print the appropriate option values

  switch optype
    case 1
      fprintf('%30s: %13d\n', optstr, ivalue);
    case 2
      fprintf('%30s: %13.4e\n', optstr, rvalue);
    case 3
      fprintf('%30s: %16s\n', optstr, cvalue);
    case 4
      fprintf('%30s: %3d  %16s\n', optstr, ivalue, cvalue);
    case 5
      fprintf('%30s: %14.4e  %16s\n', optstr, rvalue, cvalue);
  end
```

9.2 Program Results

```
     d01zk example results

              Quadrature rule: GK41
          Maximum Subdivisions:            50
                 Extrapolation: ON
       Extrapolation Safeguard:    1.0000e-12

Integral |  needi  |   dinest   |   errest
        1         0  -2.8431e-02   1.1234e-14
        2         0   7.9083e-03   2.6600e-09
```