NAG Toolbox

nag_glopt_optset (e05zk)

1 Purpose

nag_glopt_optset (e05zk) either initializes or resets the optional parameter arrays or sets a single optional parameter for supported problem solving functions in Chapter E05. The following functions are supported:

nag_glopt_bnd_pso (e05sa),

nag_glopt_nlp_pso (e05sb),

nag_glopt_nlp_multistart_sqp (e05uc),

nag_glopt_nlp_multistart_sqp_lsq (e05us).

2 Syntax

```
[iopts, opts, ifail] = nag_glopt_optset(optstr, iopts, opts, 'liopts', liopts, 'lopts', lopts)
[iopts, opts, ifail] = e05zk(optstr, iopts, opts, 'liopts', liopts, 'lopts', lopts)
```

3 Description

nag_glopt_optset (e05zk) 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_glopt_optget (e05zl). 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 optional parameter arrays to be passed to nag_glopt_bnd_pso (e05sa), nag_glopt_optset (e05zk) is called as follows:

```
[iopts, opts, ifail] = e05zk('Initialize = e05sa', iopts, opts);
```

Information relating to available option names and their corresponding valid values is given in Section 12 in nag_glopt_bnd_pso (e05sa), nag_glopt_nlp_pso (e05sb), nag_glopt_nlp_multistart_sqp (e05uc) and nag_glopt_nlp_multistart_sqp_lsq (e05us).

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 12 in nag_glopt_bnd_pso (e05sa), nag_glopt_nlp_pso (e05sb), nag_glopt_nlp_multistart_sqp (e05uc) and nag_glopt_nlp_multistart_sqp_lsq (e05us) 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_glopt_optset (e05zk), nag_glopt_optget (e05zl) 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_glopt_optset (e05zk), nag_glopt_optget (e05zl) 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 $= 11$

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

ifail $= 12$

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

ifail $= 13$

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

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

The *option* in **optstr** is associated with a numerical value.

ifail $= 14$

On entry, the *option* in **optstr** has been detected as **Initialize**, however the *optval*, $\langle value \rangle$, associated with **optstr** has not been recognized as a valid function name.

ifail $= 15$

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

ifail $= 16$

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

ifail $= 17$

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

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.

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

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

Some options have default values which are problem dependent. For example the option **Maximum Iterations Completed** for nag_glopt_bnd_pso (e05sa) has the default value $1000 \times$ **ndim**. If options such as this are set, they may only be set to constant values. If such an option is reset to its DEFAULT value its dependence on the specific problem will be restored.

9 Example

See the example programs associated with the problem solving function you wish to use for a demonstration of how to use nag_glopt_optset (e05zk) to initialize option arrays and set options.

9.1 Program Text

```
function e05zk_example

fprintf('e05zk example results\n\n');

npar = nag_int(5);
bu = [ 500,  500];
bl = [-500, -500];

x_target = [-420.9687463599820;-420.9687463599820];
f_target = -837.9657745448674;

iopts = zeros(100, 1, nag_int_name);
opts  = zeros(100, 1);

% Initialize the option arrays for e05sa
[iopts, opts, ifail] = e05zk(...
                            'Initialize = e05sa', iopts, opts);

% Query some default option values.
[ivalue, rvalue, boundary, optype, ifail] = ...
  e05zl(...
        'Boundary', iopts, opts);
[maxits, rvalue, itstr, optype, ifail] = ...
  e05zl(...
        'Maximum Iterations Completed', iopts, opts);
[ivalue, distol, cvalue, optype, ifail] = ...
  e05zl(...
        'Distance Tolerance', iopts, opts);

fprintf('\nDefault Option Queries:\n\n');
itstr = strtrim(itstr);
```

```
fprintf('Boundary                      : %s\n', boundary);
fprintf('Maximum Iterations Completed : %d (%s)\n', maxits, strtrim(cvalue));
fprintf('Distance Tolerance            : %10.4e\n', distol);

fprintf('\n1. Solution without using coupled local minimizer.\n');

% Set various options to non-default values if required.
[iopts, opts, ifail] = e05zk(...
                            'Repeatability = On', iopts, opts);
[iopts, opts, ifail] = e05zk(...
                            'Verify Gradients = Off', iopts, opts);
[iopts, opts, ifail] = e05zk(...
                            'Boundary = Hyperspherical', iopts, opts);
[iopts, opts, ifail] = e05zk(...
                            'Maximum iterations static = 150', iopts, opts);
[iopts, opts, ifail] = e05zk(...
                            'Repulsion Initialize = 30', iopts, opts);
[iopts, opts, ifail] = e05zk(...
                            'Repulsion Finalize = 30', iopts, opts);

% Catch warnings and assume ifail=1,3 give a good estimate
wstat = warning();
warning('OFF');
% Call e05sa to search for the global optimum.
[xb, fb, iopts, opts, user, itt, inform, ifail] = ...
  e05sa(...
        bl, bu, @objfun, 'e05sxm', iopts, opts, 'npar', npar);

switch ifail
  case {0,1}
    % e05sa encountered no errors during operation,
    % and will have returned the best optimum found.
    display_result(x_target, f_target, xb,fb,itt,inform);
  case 3
    % An instruction to exit was received by e05sa from objfun or monmod.
    % The exit flag will have been returned in inform.
    display_result(x_target, f_target, xb,fb,itt,inform);
  otherwise
    % An error was detected, and a warning has been displayed
end

fprintf('\n2. Solution using coupled local minimizer e04cb.\n');

% Set an objective target
optstr = sprintf('Target Objective Value = %32.16e', f_target);
[iopts, opts, ifail] = e05zk(...
                            optstr, iopts, opts);
[iopts, opts, ifail] = e05zk(...
                          'Target Objective Tolerance = 1.0e-5', iopts, opts);
[iopts, opts, ifail] = e05zk(...
                          'Target Objective Safeguard = 1.0e-8', iopts, opts);

% Set the local minimizer to be e04cb and set corresponding options
[iopts, opts, ifail] = e05zk(...
                            'Local Minimizer = e04cb', iopts, opts);
[iopts, opts, ifail] = e05zk(...
                            'Local Interior Iterations = 10', iopts, opts);
[iopts, opts, ifail] = e05zk(...
                            'Local Exterior Iterations = 20', iopts, opts);
[iopts, opts, ifail] = e05zk(...
                            'Local Interior Tolerance = 1.0e-4', iopts, opts);
[iopts, opts, ifail] = e05zk(...
                            'Local Exterior Tolerance = 1.0e-4', iopts, opts);

% Call e05sa to search for the global optimum.
[xb, fb, iopts, opts, user, itt, inform, ifail] = ...
  e05sa(...
        bl, bu, @objfun, 'e05sxm', iopts, opts, 'npar', npar);

switch ifail
  case {0,1}
```

```
    % e05sa encountered no errors during operation,
    % and will have returned the best optimum found.
    display_result(x_target, f_target, xb,fb,itt,inform);
  case 3
    % An instruction to exit was received by e05sa from objfun or monmod.
    % The exit flag will have been returned in inform.
    display_result(x_target, f_target, xb,fb,itt,inform);
  otherwise
    % An error was detected, and a warning has been displayed
end

fprintf('\n3. Solution using coupled local minimizer e04dg.\n');

% Set the local minimizer to be e04dg and set corresponding options
[iopts, opts, ifail] = e05zk(...
                            'Local Minimizer = e04dg', iopts, opts);
[iopts, opts, ifail] = e05zk(...
                            'Local Interior Iterations = 5', iopts, opts);
[iopts, opts, ifail] = e05zk(...
                            'Local Exterior Iterations = 20', iopts, opts);

%[iopts, opts, ifail] = e05zk('Verify Gradients = off', iopts, opts);
% Call e05sa to search for the global optimum.
[xb, fb, iopts, opts, user, itt, inform, ifail] = ...
      e05sa(...
            bl, bu, @objfun, 'e05sxm', iopts, opts, 'npar', npar);

switch ifail
  case {0,1}
    % e05sa encountered no errors during operation,
    % and will have returned the best optimum found.
    display_result(x_target, f_target, xb,fb,itt,inform);
  case 3
    % An instruction to exit was received by e05sa from objfun or monmod.
    % The exit flag will have been returned in inform.
    display_result(x_target, f_target, xb,fb,itt,inform);
  otherwise
    % An error was detected, and a warning has been displayed
end

warning(wstat);

function [mode, objf, vecout, user] = ...
  objfun(mode, ndim, x, objf, vecout, nstate, user)
  % Test nstate to indicate what stage of computation has been reached.
  switch nstate
    case (2)
      % objfun is called for the very first time.
    case (-1)
      % objfun is called for the first time on a new slave thread.
      % This will never happen if running in a serial context.
    case (1)
      % objfun is called on entry to a NAG local minimiser.
    case (0)
      % This will be the normal value of NSTATE.
    otherwise
      % This is extremely unlikely, and indicates that an error has
      % occurred on the system
      mode = nag_int(-1);
      error('*** Error detected in objfun');
  end

  % Test mode to determine whether to calculate objf and/or objgrd.
  evalobjf = false;
  evalobjg = false;
  switch mode
    case {0,5}
      % Only the value of the objective function is needed.
      evalobjf = true;
    case {1,6}
      % Only the values of the NDIM gradients are required.
```

```
      evalobjg = true;
    case {2,7}
      % Both the objective function and the NDIM gradients are required.
      evalobjf = true;
      evalobjg = true;
    otherwise
      mode = nag_int(-1);
      error('*** Illegal value of mode (%d) in objfun', mode);
  end

  if evalobjf
    % Evaluate the objective function.
    objf = sum(x(1:double(ndim)).*sin(sqrt(abs(x(1:double(ndim))))));
  end

  if evalobjg
    % Calculate the gradient of the objective function,
    % and return the result in vecout.
    vecout = sqrt(abs(x));
    for i=1:double(ndim)
      vi = vecout(i);
      xi = x(i);
      if abs(xi) < x02aj
        vecout(i) = 0;
      else
        vecout(i) = sin(vi) + signtransfer(xi*cos(vi)/(2.0*vi), xi);
      end
    end
  end

  function [r] = signtransfer(x, y)
    if y >= 0
      r = abs(x);
    else
      r = -abs(x);
    end

function [] = display_result(x_target, f_target, xb,fb,itt,inform)

  % Display final counters.
  fprintf('\nAlgorithm Statistics\n--------------------\n');
  fprintf('Total complete iterations             : %4d\n', itt(1));
  fprintf('Complete iterations since improvement : %4d\n', itt(2));
  fprintf('Total particles converged to xb       : %4d\n', itt(3));
  fprintf('Total improvements to global optimum  : %4d\n', itt(4));
  fprintf('Total function evaluations            : %4d\n', itt(5));
  fprintf('Total particles re-initialized        : %4d\n\n', itt(6));

  % Display why finalization occurred.
  switch inform
  case 0
     fprintf('Solution Status : An error was detected by e05sa\n');
  case 1
     fprintf('Solution Status : Target value achieved\n');
  case 2
     fprintf('Solution Status : Minimum swarm standard deviation obtained\n');
  case 3
     fprintf('Solution Status : Sufficient particles converged\n');
  case 4
     fprintf('Solution Status : No improvement in preset iteration limit\n');
  case 5
     fprintf('Solution Status : Maximum complete iterations attained\n');
  case 6
     fprintf('Solution Status : Maximum function evaluations exceeded\n');
  otherwise
     fprintf('User termination case:  %d\n', inform);
  end

  % Display final objective value and location.
  fprintf('\n  Known objective optimum  : %13.5f\n', f_target);
  fprintf('  Achieved objective value : %13.5f\n\n', fb);
```

```
  title = 'Comparison between known and achieved optima.';
  clabs = {'x_target'; 'xb      '};
  ncols = nag_int(80);
  indent = nag_int(0);
  [ifail] = x04cb(...
                 'G', 'N', [x_target, xb], 'f9.2', title, 'I', clabs, ...
                 'C', clabs, ncols, indent);
  fprintf('\n');
```

9.2 Program Results

```
     e05zk example results

Default Option Queries:

Boundary                     : FLOATING
Maximum Iterations Completed : 1000 ()
Distance Tolerance           : 1.0000e-04

1. Solution without using coupled local minimizer.

Algorithm Statistics
--------------------
Total complete iterations             :  395
Complete iterations since improvement :  152
Total particles converged to xb       :    2
Total improvements to global optimum  :   59
Total function evaluations            : 2773
Total particles re-initialized        :    2

Solution Status : No improvement in preset iteration limit

  Known objective optimum  :    -837.96577
  Achieved objective value :    -837.96567

 Comparison between known and achieved optima.
    x_target       xb
 1   -420.97  -420.95
 2   -420.97  -420.94

2. Solution using coupled local minimizer e04cb.

Algorithm Statistics
--------------------
Total complete iterations             :   51
Complete iterations since improvement :    1
Total particles converged to xb       :    0
Total improvements to global optimum  :   12
Total function evaluations            :  537
Total particles re-initialized        :    0

Solution Status : Target value achieved

  Known objective optimum  :    -837.96577
  Achieved objective value :    -837.96577

 Comparison between known and achieved optima.
    x_target       xb
 1   -420.97  -420.97
 2   -420.97  -420.97

3. Solution using coupled local minimizer e04dg.

Algorithm Statistics
--------------------
Total complete iterations             :    8
Complete iterations since improvement :    1
Total particles converged to xb       :    0
Total improvements to global optimum  :   10
```

```
Total function evaluations            :  120
Total particles re-initialized        :    0

Solution Status : Target value achieved

  Known objective optimum  :    -837.96577
  Achieved objective value :    -837.96561

 Comparison between known and achieved optima.
    x_target       xb
 1   -420.97  -420.94
 2   -420.97  -420.98
```