1 SUMMARY

Given real n by n symmetric matrices $\mathbf{H}$ and $\mathbf{M}$ (with $\mathbf{M}$ positive definite), real n vectors $\mathbf{c}$ and $\mathbf{o}$, and scalars $\sigma \geq 0$, $\varepsilon \geq 0$ and f_0, this package finds an **approximate minimizer of the regularised objective function**

$$\tfrac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{c}^T\mathbf{x} + f_0 + \tfrac{1}{p}\sigma[\|\mathbf{x}+\mathbf{o}\|_{\mathbf{M}}^2 + \varepsilon]^{p/2},$$

where $\|\mathbf{v}\|_{\mathbf{M}} = \sqrt{\mathbf{v}^T\mathbf{M}\mathbf{v}}$ is the $\mathbf{M}$-norm of $\mathbf{v}$. This problem commonly occurs as a subproblem in nonlinear optimization calculations involving cubic regularisation. The method may be suitable for large n as no factorization of $\mathbf{H}$ is required. Reverse communication is used to obtain matrix-vector products of the form $\mathbf{H}\mathbf{z}$, $\mathbf{M}^{-1}\mathbf{z}$ and, perhaps, $\mathbf{M}\mathbf{z}$.

ATTRIBUTES — Versions: `GALAHAD_GLRT_single`, `GALAHAD_GLRT_double`. **Uses:** `GALAHAD_SYMBOLS`, `GALA-HAD_SPACE`, `GALAHAD_RAND`, `GALAHAD_NORMS`, `GALAHAD_GLTR`, `GALAHAD_ROOTS`, `GALAHAD_SPECFILE`, `*TTRF`. **Date:** November 2007. **Origin:** N. I. M. Gould, Oxford University and Rutherford Appleton Laboratory. **Language:** Fortran 95 + TR 15581 or Fortran 2003.

2 HOW TO USE THE PACKAGE

Access to the package requires a `USE` statement such as

Single precision version

        USE GALAHAD_GLRT_single

Double precision version

        USE GALAHAD_GLRT_double

If it is required to use both modules at the same time, the derived types `GLRT_control_type`, `GLRT_inform_type`, `GLRT_data_type`, (Section 2.1) and the subroutines `GLRT_initialize`, `GLRT_solve`, `GLRT_terminate` (Section 2.2) and `GLRT_read_specfile` (Section 2.6) must be renamed on one of the `USE` statements.

2.1 The derived data types

Three derived data types are accessible from the package.

2.1.1 The derived data type for holding control parameters

The derived data type `GLRT_control_type` is used to hold controlling data. Default values may be obtained by calling `GLRT_initialize` (see Section 2.2.1). The components of `GLRT_control_type` are:

`error` is a scalar variable of type default `INTEGER`, that holds the stream number for error messages. Printing of error messages in `GLRT_solve` and `GLRT_terminate` is suppressed if `error` $\leq$ `0`. The default is `error = 6`.

`out` is a scalar variable of type default `INTEGER`, that holds the stream number for informational messages. Printing of informational messages in `GLRT_solve` is suppressed if `out` < `0`. The default is `out = 6`.

print_level is a scalar variable of type default INTEGER, that is used to control the amount of informational output which is required. No informational output will occur if print_level ≤ 0. If print_level = 1 a single line of output will be produced for each iteration of the process. If print_level ≥ 2 this output will be increased to provide significant detail of each iteration. The default is print_level = 0.

itmax is a scalar variable of type default INTEGER, that holds the maximum number of iterations which will be allowed in GLRT_solve. If itmax is set to a negative number, it will be reset by GLRT_solve to n. The default is itmax = -1.

extra_vectors is a scalar variable of type default INTEGER, that specifies the number of additional vectors of length n that will be allocated to try to speed up the computation during the second pass. The default is extra_vectors = 0.

stopping_rule is a scalar variable of type default INTEGER, that flags the stopping rule to be used (see stop_relative and stop_absolute below). Appropriate values are in the range $[0,2]$, and any value outside this range will be interpreted as 0. The default is stopping_rule = 0.

freq is a scalar variable of type default INTEGER, that defines the frequency at which the tridiagonal subproblem will be solved. Specifically, the subproblem will be solved on iterations $k = 1 + \text{freq} * i$, for $i = 1, 2, \ldots$. If itmax is ≤ 1, it will be reset by GLRT_solve to 1. The default is freq = 1.

unitm is a scalar variable of type default LOGICAL, that must be set .TRUE. if the matrix $\mathbf{M}$ is the identity matrix, and .FALSE. otherwise. The default is unitm = .TRUE..

space_critical is a scalar variable of type default LOGICAL, that may be set .TRUE. if the user wishes the package to allocate as little internal storage as possible, and .FALSE. otherwise. The package may be more efficient if space_critical is set .FALSE.. The default is space_critical = .FALSE..

deallocate_error_fatal is a scalar variable of type default LOGICAL, that may be set .TRUE. if the user wishes the package to return to the user in the unlikely event that an internal array deallocation fails, and .FALSE. if the package should be allowed to try to continue. The default is deallocate_error_fatal = .FALSE..

stop_relative and stop_absolute are scalar variables of type REAL (double precision in GALAHAD_GLRT_double), that holds the relative and absolute convergence tolerances (see Section 4). The computed solution $\mathbf{x}$ is accepted by GLRT_solve if the computed value of $\|\mathbf{Hx} + \lambda\mathbf{Mx} + \mathbf{c}\|_{\mathbf{M}^{-1}}$ is less than or equal to $\max(\nu\|\mathbf{c}\|_{\mathbf{M}^{-1}} *$ stop_relative, stop_absolute), where $\lambda = \sigma[\|\mathbf{x} + \mathbf{o}\|_{\mathbf{M}}^2 + \epsilon]^{p/2-1}$ and ν depends on the stopping rule selected by stopping_rule: for stopping_rule = 0, $\nu = 1$, for stopping_rule = 1, $\nu = \min(1, \|\mathbf{x}\|)$, and for stopping_rule = 2, $\nu = \min(1, \|\mathbf{x}\|/\max(1, \sigma))$. The defaults are stop_relative = $\sqrt{u}$ and stop_absolute = 0.0, where u is EPSILON(1.0) (EPSILON(1.0D0) in GALAHAD_GLRT_double).

fraction_opt is a scalar variable of type default REAL (double precision in GALAHAD_GLRT_double), that specifies the fraction of the optimal value which is to be considered acceptable by the algorithm. A negative value is considered to be zero, and a value of larger than one is considered to be one. Reducing fraction_opt below one will result in a reduction of the computation performed at the expense of an inferior optimal value. The default is fraction_opt = 1.0.

f_0 is a scalar variable of type default REAL (double precision in GALAHAD_GLRT_double), that gives the value of the constant term f_0 in the objective function. This value has no effect on the computed minimizer $\mathbf{x}$. The default is f_0 = 0.0.

rminvr_zero is a scalar variable of type default REAL (double precision in GALAHAD_GLRT_double), that gives the smallest value that the square of the $\mathbf{M}$-norm of the gradient of the objective function may be before it is considered to be zero. The default is rminvr_zero = 10 u, where u is EPSILON(1.0) (EPSILON(1.0D0) in GALAHAD_GLRT_double).

`prefix` is a scalar variable of type default `CHARACTER` and length 30, that may be used to provide a user-selected character string to preface every line of printed output. Specifically, each line of output will be prefaced by the string `prefix(2:LEN(TRIM(prefix))-1)`, thus ignoring the first and last non-null components of the supplied string. If the user does not want to preface lines by such a string, they may use the default `prefix = ""`.

2.1.2 The derived data type for holding informational parameters

The derived data type `GLRT_inform_type` is used to hold parameters that give information about the progress and needs of the algorithm. The components of `GLRT_inform_type` are:

`status` is a scalar variable of type default `INTEGER`, that gives the current status of the algorithm. See Sections 2.3 and 2.4 for details.

`alloc_status` is a scalar variable of type default `INTEGER`, that gives the status of the last internal array allocation or deallocation. This will be 0 if `status = 0`.

`bad_alloc` is a scalar variable of type default `CHARACTER` and length 80, that gives the name of the last internal array for which there were allocation or deallocation errors. This will be the null string if `status = 0`.

`obj` is a scalar variable of type default `REAL` (double precision in `GALAHAD_GLRT_double`), that holds the value of the quadratic function $\frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{c}^T\mathbf{x} + f_0$.

`obj_regularized` is a scalar variable of type default `REAL` (double precision in `GALAHAD_GLRT_double`), that holds the value of the regularized objective function $\frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{c}^T\mathbf{x} + f_0 + \frac{1}{p}\sigma[\|\mathbf{x}+\mathbf{o}\|_{\mathbf{M}}^2 + \varepsilon]^{p/2}$.

`multiplier` is a scalar variable of type default `REAL` (double precision in `GALAHAD_GLRT_double`), that holds the value of the multiplier $\lambda = \sigma[\|\mathbf{x}+\mathbf{o}\|_{\mathbf{M}}^2 + \varepsilon]^{p/2-1}$.

`leftmost` is a scalar variable of type default `REAL` (double precision in `GALAHAD_GLRT_double`), that holds an estimate of the leftmost eigenvalue of the matrix pencil $(\mathbf{H}, \mathbf{M})$.

`iter` is a scalar variable of type default `INTEGER`, that holds the current number of Lanczos vectors used.

`iter_pass2` is a scalar variable of type default `INTEGER`, that holds the current number of Lanczos vectors used in the second pass.

`negative_curvature` is a scalar variable of type default `LOGICAL`, that is set `.TRUE.` if $\mathbf{H}$ has been found to be indefinite during the calculation and `.FALSE.` otherwise.

2.1.3 The derived data type for holding problem data

The derived data type `GLRT_data_type` is used to hold all the data for a particular problem between calls of `GLRT` procedures. This data should be preserved, untouched, from the initial call to `GLRT_initialize` to the final call to `GLRT_terminate`.

2.2 Argument lists and calling sequences

There are three procedures for user calls (see Section 2.6 for further features):

1. The subroutine `GLRT_initialize` is used to set default values, and initialize private data.

2. The subroutine `GLRT_solve` is called repeatedly to solve the problem. On each exit, the user may be expected to provide additional information and, if necessary, re-enter the subroutine.

3. The subroutine `GLRT_terminate` is provided to allow the user to automatically deallocate array components of the private data, allocated by `GLRT_solve`, at the end of the solution process. It is important to do this if the data object is re-used for another problem since `GLRT_initialize` cannot test for this situation, and any existing associated targets will subsequently become unreachable.

We use square brackets `[ ]` to indicate `OPTIONAL` arguments.

2.2.1 The initialization subroutine

Default values are provided as follows:

```
CALL GLRT_initialize( data, control, inform )
```

data is a scalar `INTENT(INOUT)` argument of type `GLRT_data_type` (see Section 2.1.3). It is used to hold data about the problem being solved.

control is a scalar `INTENT(OUT)` argument of type `GLRT_control_type` (see Section 2.1.1). On exit, `control` contains default values for the components as described in Section 2.1.1. These values should only be changed after calling `GLRT_initialize`.

inform is a scalar `INTENT(OUT)` argument of type `GLRT_inform_type` (see Section 2.1.2). A successful call to GL-RT_initialize is indicated when the component `status` has the value 0. For other return values of `status`, see Section 2.4.

2.2.2 The optimization problem solution subroutine

The optimization problem solution algorithm is called as follows:

```
CALL GLRT_solve( n, p, sigma, X, R, VECTOR, data, control, inform[, eps, O] )
```

n is a scalar `INTENT(IN)` argument of type default `INTEGER`, that must be set to the number of unknowns, n. **Restriction:** $n > 0$.

p is a scalar `INTENT(IN)` variable of type default `REAL` (double precision in `GALAHAD_GLRT_double`), that must be set on initial entry to the desired order p of regularisation. **Restriction:** $p \geq 2$.

sigma is a scalar `INTENT(IN)` variable of type default `REAL` (double precision in `GALAHAD_GLRT_double`), that must be set on initial entry to the value of the weight σ associated with the regularisation term. **Restriction:** $\sigma \geq 0$.

X is an array `INTENT(INOUT)` argument of dimension n and type default `REAL` (double precision in `GALAHAD_GLRT_double`), that holds an estimate of the solution **x** of the linear system. On initial entry, X need not be set. It must not be changed between entries. On exit, X contains the current best estimate of the solution.

R is an array `INTENT(INOUT)` argument of dimension n and type default `REAL` (double precision in `GALAHAD_GLRT_double`), that is used to hold the gradient **Hx** + **c** of the objective function at the current estimate of the solution. On initial entry, R must contain the vector **c**. If `inform%status` = 5 on exit, it must be reset to **c**; otherwise it must be left unchanged. On exit, R contains the gradient of the objective function at the current best estimate of the solution.

VECTOR is an array `INTENT(INOUT)` argument of dimension n and type default `REAL` (double precision in `GALAHAD_GLRT_double`), that is used to pass information from and to `GLRT_solve`, as explained in Section 2.3. On initial entry, VECTOR need not be set. On exit, the actual content of the array depends on the value of the parameter `inform%status` (see Section 2.3).

data is a scalar `INTENT(INOUT)` argument of type `GLRT_data_type` (see Section 2.1.3). It is used to hold data about the problem being solved. It must not have been altered **by the user** since the last call to `GLRT_initialize`.

---

control is a scalar `INTENT(IN)` argument of type `GLRT_control_type`. (see Section 2.1.1). Default values may be assigned by calling `GLRT_initialize` prior to the first call to `GLRT_solve`.

inform is a scalar `INTENT(INOUT)` argument of type `GLRT_inform_type` (see Section 2.1.2). On initial entry, the component `status` must be set to 1. The remaining components need not be set. A successful call to `GLRT_solve` is indicated when the component `status` has the value 0. For other return values of `status`, see Sections 2.3 and 2.4.

eps is an optional scalar `INTENT(IN)` variable of type default `REAL` (double precision in `GALAHAD_GLRT_double`). If `eps` is `PRESENT`, it must be set on initial entry to the value of the shift ε associated with the regularisation term. **Restriction:** $\varepsilon \geq 0$.

O is an optional array `INTENT(IN)` variable of type default `REAL` (double precision in `GALAHAD_GLRT_double`). If `O` is `PRESENT`, it must be set on initial entry to the value of the offset o associated with the regularisation term.

2.2.3 The termination subroutine

All previously allocated arrays are deallocated as follows:

        `CALL GLRT_terminate( data, control, inform )`

data is a scalar `INTENT(INOUT)` argument of type `GLRT_data_type` exactly as for `GLRT_solve` that must not have been altered **by the user** since the last call to `GLRT_initialize`. On exit, array components will have been deallocated.

control is a scalar `INTENT(IN)` argument of type `GLRT_control_type` exactly as for `GLRT_solve`.

inform is a scalar `INTENT(OUT)` argument of type `GLRT_type` exactly as for `GLRT_solve`. Only the component `status` will be set on exit, and a successful call to `GLRT_terminate` is indicated when this component `status` has the value 0. For other return values of `status`, see Section 2.4.

2.3 Reverse communication

A positive value of `inform%status` on exit from `GLRT_solve` indicates that the user needs to take appropriate action before re-entering the subroutine. Possible values are:

2. The user must perform the preconditioning operation

$$\mathbf{y} := \mathbf{M}^{-1}\mathbf{z},$$

and recall `GLRT_solve`. The vector $\mathbf{z}$ is available in the array `VECTOR`, and the result $\mathbf{y}$ must be placed in `VECTOR`. No argument except `VECTOR` should be altered before recalling `GLRT_solve`. This return can only occur when `control%unitm` is `.FALSE.`.

3. The user must perform the matrix-vector product

$$\mathbf{y} := \mathbf{Hz}$$

and recall `GLRT_solve`. The vector $\mathbf{z}$ is available in the array `VECTOR`, and the result $\mathbf{y}$ must be placed in `VECTOR`. No argument except `VECTOR` should be altered before recalling `GLRT_solve`.

4. The user should reset `R` to $\mathbf{c}$ and recall `GLRT_solve`. No argument except `R` should be altered before recalling `GLRT_solve`.

5. The user must perform the operation

$$\mathbf{y} := \mathbf{Mz},$$

and recall `GLRT_solve`. The vector $\mathbf{z}$ is available in the array `VECTOR`, and the result $\mathbf{y}$ must be placed in `VECTOR`. No argument except `VECTOR` should be altered before recalling `GLRT_solve`. This return can only occur when both `control%unitm` is `.FALSE.` and `O` is `PRESENT`.

All use is subject to licence. See `http://galahad.rl.ac.uk/galahad-www/cou.html`.
For any commercial application, a separate license must be signed.

2.4 Warning and error messages

A negative value of `inform%status` on exit from `GLRT_solve` or `GLRT_terminate` indicates that an error has occurred. No further calls should be made until the error has been corrected. Possible values are:

-1. An allocation error occurred. A message indicating the offending array is written on unit `control%error`, and the returned allocation status and a string containing the name of the offending array are held in `inform%alloc_-status` and `inform%bad_alloc` respectively.

-2. A deallocation error occurred. A message indicating the offending array is written on unit `control%error` and the returned allocation status and a string containing the name of the offending array are held in `inform%alloc_-status` and `inform%bad_alloc` respectively.

-3. (`GLRT_solve` only) One or more of the restrictions `n > 0`, `sigma` ≥ 0, `eps` ≥ 0 or `p` ≥ 2 has been violated.

-7. (`GLRT_solve` only) The problem is unbounded from below. This can only happen if $p \leq 2$. In this case, the problem is unbounded along the arc `X` $+\alpha$ `VECTOR` as α increases.

-15. (`GLRT_solve` only) The matrix **M** appears not to be positive definite.

-18. (`GLRT_solve` only) More than `control%itmax` iterations have been performed without obtaining convergence.

-25. (`GLRT_solve` only) `inform%status` is not > 0 on entry.

2.5 Re-entry with a new value of σ

It commonly happens that, having solved the problem for a particular value of the weight σ, a user now wishes to solve the problem for a different value of σ. Rather than restarting the calculation with `inform%status = 1`, a useful approximation may be found resetting `sigma` to the new required value and R to **c**, and recalling `GLRT_solve` with `inform%status = 6` and the remaining arguments unchanged. This will determine the best solution within the Krylov space investigated in the previous minimization (see Section 4).

2.6 Further features

In this section, we describe an alternative means of setting control parameters, that is components of the variable `control` of type `GLRT_control_type` (see Section 2.1.1), by reading an appropriate data specification file using the subroutine `GLRT_read_specfile`. This facility is useful as it allows a user to change `GLRT` control parameters without editing and recompiling programs that call `GLRT`.

A specification file, or specfile, is a data file containing a number of "specification commands". Each command occurs on a separate line, and comprises a "keyword", which is a string (in a close-to-natural language) used to identify a control parameter, and an (optional) "value", which defines the value to be assigned to the given control parameter. All keywords and values are case insensitive, keywords may be preceded by one or more blanks but values must not contain blanks, and each value must be separated from its keyword by at least one blank. Values must not contain more than 30 characters, and each line of the specfile is limited to 80 characters, including the blanks separating keyword and value.

The portion of the specification file used by `GLRT_read_specfile` must start with a "BEGIN GLRT" command and end with an "END" command. The syntax of the specfile is thus defined as follows:

```
( .. lines ignored by GLRT_read_specfile .. )
  BEGIN GLRT
     keyword    value
     .......    .....
     keyword    value
```

All use is subject to licence. See `http://galahad.rl.ac.uk/galahad-www/cou.html`**.**
For any commercial application, a separate license must be signed.

```
      END
( .. lines ignored by GLRT_read_specfile .. )
```

where keyword and value are two strings separated by (at least) one blank. The "BEGIN GLRT" and "END" delimiter command lines may contain additional (trailing) strings so long as such strings are separated by one or more blanks, so that lines such as

```
    BEGIN GLRT SPECIFICATION
```

and

```
    END GLRT SPECIFICATION
```

are acceptable. Furthermore, between the "BEGIN GLRT" and "END" delimiters, specification commands may occur in any order. Blank lines and lines whose first non-blank character is ! or * are ignored. The content of a line after a ! or * character is also ignored (as is the ! or * character itself). This provides an easy manner to "comment out" some specification commands, or to comment specific values of certain control parameters.

The value of a control parameters may be of three different types, namely integer, logical or real. Integer and real values may be expressed in any relevant Fortran integer and floating-point formats (respectively). Permitted values for logical parameters are "ON", "TRUE", ".TRUE.", "T", "YES", "Y", or "OFF", "NO", "N", "FALSE", ".FALSE." and "F". Empty values are also allowed for logical control parameters, and are interpreted as "TRUE".

The specification file must be open for input when GLRT_read_specfile is called, and the associated device number passed to the routine in device (see below). Note that the corresponding file is REWINDed, which makes it possible to combine the specifications for more than one program/routine. For the same reason, the file is not closed by GLRT_read_specfile.

2.6.1   To read control parameters from a specification file

Control parameters may be read from a file as follows:

```
    CALL GLRT_read_specfile( control, device )
```

control is a scalar INTENT(INOUT) argument of type GLRT_control_type (see Section 2.1.1). Default values should have already been set, perhaps by calling GLRT_initialize. On exit, individual components of control may have been changed according to the commands found in the specfile. Specfile commands and the component (see Section 2.1.1) of control that each affects are given in Table 2.1.

device is a scalar INTENT(IN) argument of type default INTEGER, that must be set to the unit number on which the specfile has been opened. If device is not open, control will not be altered and execution will continue, but an error message will be printed on unit control%error.

2.7   Information printed

If control%print_level is positive, information about the progress of the algorithm will be printed on unit control%out. If control%print_level = 1, a single line of output will be produced for each iteration of the process. This will include the iteration number, the value of the objective function, the $\mathbf{M}^{-1}$-norm of its gradient $\|(\mathbf{H} + \lambda\mathbf{M})\mathbf{x} + \mathbf{c}\|_{\mathbf{M}^{-1}}$, the value of the multiplier $\lambda = \sigma[\|\mathbf{x} + \mathbf{o}\|_{\mathbf{M}}^2 + \varepsilon]^{p/2-1}$, the number of Newton steps required to find λ, and the exit code from this calculation (0 = successful, 1 = 1, stalled, 2 = more than 100 steps). If control%print_level $\geq$ 2, this output will be increased to provide significant detail of each iteration. This extra output includes a complete history of the inner iteration required to solve the "tridiagonal" subproblem, and for each Newton iteration records the estimate of λ, the error $\theta(\lambda) = [\|\mathbf{x} + \mathbf{o}\|_{\mathbf{M}}^2 + \varepsilon]^{p/2-1} - \lambda/\sigma$ and the $\mathbf{M}$-norm of $\mathbf{x}$.

---

command	component of `control`	value type
`error-printout-device`	`%error`	integer
`printout-device`	`%out`	integer
`print-level`	`%print_level`	integer
`maximum-number-of-iterations`	`%itmax`	integer
`number-extra-n-vectors-used`	`%extra_vectors`	integer
`stopping-rule`	`%stopping_rule`	integer
`tri-diagonal-solve-frequency`	`%freq`	integer
`relative-accuracy-required`	`%stop_relative`	real
`absolute-accuracy-required`	`%stop_absolute`	real
`fraction-optimality-required`	`%fraction_opt`	real
`constant-term-in-objective`	`%f_0`	real
`zero-gradient-tolerance`	`%rminvr_zero`	real
`two-norm-regularisation`	`%unitm`	logical
`space-critical`	`%space_critical`	logical
`deallocate-error-fatal`	`%deallocate_error_fatal`	logical

Table 2.1: Specfile commands and associated components of `control`.

3   GENERAL INFORMATION

Use of common: None.

Workspace: Provided automatically by the module.

Other routines called directly: `GLRT_solve` calls the LAPACK subroutine `*PTTRF`, where `*` is `S` for the default real version and `D` for the double precision version.

Other modules used directly: `GLRT_solve` calls the GALAHAD packages `GALAHAD_SYMBOLS`, `GALAHAD_SPACE`, `GA-LAHAD_RAND`, `GALAHAD_NORMS`, `GALAHAD_GLTR`, `GALAHAD_ROOTS` and `GALAHAD_SPECFILE`.

Input/output: Output is under control of the arguments `control%error`, `control%out` and `control%print_level`.

Restrictions: $n > 0$, $\sigma > 0$.

Portability: ISO Fortran 95 + TR 15581 or Fortran 2003. The package is thread-safe.

4   METHOD

The required solution $\mathbf{x}$ necessarily satisfies the optimality condition $\mathbf{Hx} + \lambda \mathbf{Mx} + \mathbf{c} + \lambda \mathbf{o} = 0$, where $\lambda = \sigma[\|\mathbf{x} + \mathbf{o}\|_{\mathbf{M}}^2 + \epsilon]^{p/2-1}$. In addition, the matrix $\mathbf{H} + \lambda \mathbf{M}$ will be positive semi-definite.

The method is iterative. Starting with the vector $\mathbf{M}^{-1}\mathbf{c}$, a matrix of Lanczos vectors is built one column at a time so that the k-th column is generated during iteration k. These columns span a so-called Krylov space. The resulting n by k matrix $\mathbf{Q}_k$ has the property that $\mathbf{Q}_k^T \mathbf{H} \mathbf{Q}_k = \mathbf{T}_k$, where $\mathbf{T}_k$ is tridiagonal. An approximation to the required solution may then be expressed formally as

$$\mathbf{x}_{k+1} = \mathbf{Q}_k \mathbf{y}_k$$

where $\mathbf{y}_k$ solves the "tridiagonal" subproblem of minimizing

$$\tfrac{1}{2}\mathbf{y}^T \mathbf{T}_k \mathbf{y} + \|\mathbf{c}\|_{\mathbf{M}^{-1}} \mathbf{e}_1^T \mathbf{y} + \tfrac{1}{p}\sigma[\|\mathbf{y} + \mathbf{d}\|_2^2 + \epsilon]^{p/2}, \tag{4.1}$$

where $\mathbf{d} = \mathbf{Q}_k^T \mathbf{M} \mathbf{o}$ and $\mathbf{e}_1$ is the first unit vector.

To minimize (4.1), the optimality conditions

$$(\mathbf{T}_k + \lambda \mathbf{I})\mathbf{y}(\lambda) = -\mathbf{c} - \lambda \mathbf{d}, \tag{4.2}$$

where $\lambda = \sigma[\|\mathbf{y}(\lambda) + \mathbf{d}\|_{\mathbf{M}}^2 + \epsilon]^{p/2-1}$ are used as the basis of an iteration. Specifically, given an estimate λ for which $\mathbf{T}_k + \lambda \mathbf{I}$ is positive definite, the tridiagonal system (4.2) may be efficiently solved to give $\mathbf{y}(\lambda)$. It is then simply a matter of adjusting λ (for example by a Newton-like process) to solve the scalar nonlinear equation

$$\theta(\lambda) \equiv [\|\mathbf{y}(\lambda) + \mathbf{d}\|_{\mathbf{M}}^2 + \epsilon]^{p/2-1} - \frac{\lambda}{\sigma} = 0. \tag{4.3}$$

In practice (4.3) is reformulated, and a more rapidly converging iteration is used.

It is possible to measure the optimality measure $\|\mathbf{Hx} + \lambda \mathbf{Mx} + \mathbf{c} + \lambda \mathbf{o}\|_{\mathbf{M}^{-1}}$ without computing $\mathbf{x}_{k+1}$, and thus without needing $\mathbf{Q}_k$. Once this measure is sufficiently small, a second pass is required to obtain the estimate $\mathbf{x}_{k+1}$ from $\mathbf{y}_k$. As this second pass is an additional expense, a record is kept of the optimal objective function values for each value of k, and the second pass is only performed so far as to ensure a given fraction of the final optimal objective value. Large savings may be made in the second pass by choosing the required fraction to be significantly smaller than one.

Special code is used in the special case $p = 2$, as in this case a single pass suffices.

Reference: The method is described in detail in

C. Cartis, N. I. M. Gould and Ph. L. Toint, Adaptive cubic regularisation methods for unconstrained optimization. Part I: motivation, convergence and numerical results. *Mathematical Programming* **127(2)**, pp.245-295, 2011.

5 EXAMPLE OF USE

Suppose we wish to solve a problem in 10,000 unknowns, whose data is

$$\mathbf{H} = \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & . & & \\ & . & . & . & \\ & & . & -2 & 1 \\ & & & 1 & -2 \end{pmatrix}, \ \mathbf{M} = \begin{pmatrix} 2 & & & & \\ & 2 & & & \\ & & . & & \\ & & & 2 & \\ & & & & 2 \end{pmatrix}, \ \mathbf{c} = \begin{pmatrix} 1 \\ 1 \\ . \\ 1 \\ 1 \end{pmatrix} \text{ and } \mathbf{o} = - \begin{pmatrix} 1 \\ 1 \\ . \\ 1 \\ 1 \end{pmatrix},$$

with a weight $\sigma = 10$ and shift $\epsilon = 1$. Suppose further that we are content with an approximation which is within 99% of the best. Then we may use the following code

```
PROGRAM GALAHAD_GLRT_EXAMPLE  !  GALAHAD 2.7 - 08/02/2016 AT 09:50 GMT.
USE GALAHAD_GLRT_DOUBLE                    ! double precision version
IMPLICIT NONE
INTEGER, PARAMETER :: working = KIND( 1.0D+0 ) ! set precision
REAL ( KIND = working ), PARAMETER :: one = 1.0_working, two = 2.0_working
INTEGER, PARAMETER :: n = 10000                ! problem dimension
INTEGER :: i
REAL ( KIND = working ) :: p = 3.0_working     ! order of regulatisation
REAL ( KIND = working ) :: eps = 1.0_working   ! shift
REAL ( KIND = working ) :: sigma = 10.0_working ! regulatisation weight
REAL ( KIND = working ), DIMENSION( n ) :: X, R, VECTOR, H_vector, O
TYPE ( GLRT_data_type ) :: data
TYPE ( GLRT_control_type ) :: control
TYPE ( GLRT_inform_type ) :: inform
CALL GLRT_initialize( data, control, inform ) ! Initialize control parameters
```

```
      control%unitm = .FALSE.           ! M is not the identity matrix
      control%fraction_opt = 0.99       ! Only require 99% of the best
      R = one                           ! The linear term c is a vector of ones
      O = - one                         ! The offset o is a vector of minus ones
      inform%status = 1
      DO                                    !  Iteration to find the minimizer
        CALL GLRT_solve( n, p, sigma, X, R, VECTOR, data, control, inform,        &
                         eps = eps, O = O )
        SELECT CASE( inform%status )      ! Branch as a result of inform%status
        CASE( 2 )                         ! Form the preconditioned vector
          VECTOR = VECTOR / two           ! Preconditioner is two times identity
        CASE ( 3 )                        ! Form the matrix-vector product
          H_vector( 1 ) = - two * VECTOR( 1 ) + VECTOR( 2 )
          DO i = 2, n - 1
            H_vector( i ) = VECTOR( i - 1 ) - two * VECTOR( i ) + VECTOR( i + 1 )
          END DO
          H_vector( n ) = VECTOR( n - 1 ) - two * VECTOR( n )
          VECTOR = H_vector
        CASE ( 4 )        !  Restart
          R = one         !  set r to c
        CASE( 5 )                         ! Form the product of the preconditioner
          VECTOR = two * VECTOR           ! with a vector
        CASE ( 0 )  !  Successful return
          WRITE( 6, "( 1X, I0, ' 1st pass and ', I0, ' 2nd pass iterations' )" )  &
            inform%iter, inform%iter_pass2
          H_vector( 1 ) = - two * X( 1 ) + X( 2 )
          DO i = 2, n - 1
            H_vector( i ) = X( i - 1 ) - two * X( i ) + X( i + 1 )
          END DO
          H_vector( n ) = X( n - 1 ) - two * X( n )
          WRITE( 6, "( ' objective recurred and calculated = ', 2ES16.8 )" )      &
           inform%obj_regularized, 0.5_working * DOT_PRODUCT( X, H_vector ) +      &
             SUM( X ) + ( sigma / p ) * ( two * DOT_PRODUCT( X + O, X + O ) +      &
               eps ) ** ( p/two )
          CALL GLRT_terminate( data, control, inform ) ! delete internal workspace
          EXIT
        CASE DEFAULT      ! Error returns
          WRITE( 6, "( ' GLRT_solve exit status = ', I6 ) " ) inform%status
          CALL GLRT_terminate( data, control, inform ) ! delete internal workspace
          EXIT
        END SELECT
      END DO
      END PROGRAM GALAHAD_GLRT_EXAMPLE
```

This produces the following output:

```
 3 1st pass and 1 2nd pass iterations
 objective recurred and calculated =   9.88721600E+03   9.88721600E+03
```