1 SUMMARY

The package uses **the preconditioned conjugate gradient method to solve the $n \times n$ symmetric positive definite linear system**

$$\mathbf{Au} = \mathbf{b}, \tag{1.1}$$

and implements several stopping criteria based on lower and upper bounds of the A-norm of the error. If $\mathbf{M} = \mathbf{U}^T\mathbf{U}$ is the preconditioning matrix, the routine actually solves the preconditioned system

$$\overline{\mathbf{A}}\mathbf{y} = \overline{\mathbf{b}}, \tag{1.2}$$

with $\overline{\mathbf{A}} = \mathbf{U}^{-T}\mathbf{A}\mathbf{U}^{-1}$ and $\overline{\mathbf{b}} = \mathbf{U}^{-T}\mathbf{b}$ and recovers the solution $\mathbf{u} = \mathbf{U}^{-1}\mathbf{y}$.

Reverse communication is used. Control is returned to the user for preconditioning operations and the products of $\mathbf{A}$ with a vector $\mathbf{z}$.

ATTRIBUTES — Version: 1.2.1 (30 January 2023). **Types:** Real (single, double). **Calls:** `_DOT`, `_NRM2`, `_SCAL`, `_AXPY`. **Date:** July 2004. **Origin:** M. Arioli, Rutherford Appleton Laboratory, and Gianmarco Manzini, IMATI-CNR, Pavia, Italy. **Language:** Fortran 90.

2 HOW TO USE THE PACKAGE

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

Single precision version

```
USE HSL_MI31_single
```

Double precision version

```
USE HSL_MI31_double
```

2.1 Argument lists and calling sequences

There are two procedures for user calls:

(a) `MI31I/ID` sets default values for the components of the arrays that hold controlling parameters. It should normally be called once prior to calls to `MI31A/AD`.

(b) `MI31A/AD` uses the conjugate gradient method to solve $\mathbf{Au} = \mathbf{b}$, optionally using preconditioning. The subroutine uses a reverse communication mechanism for preconditioning operations and matrix-vector products.

2.1.1 To set default values of controlling parameters

The single precision version

```
CALL MI31I(ICNTL, CNTL)
```

The double precision version

```
CALL MI31ID(ICNTL, CNTL)
```

`ICNTL` is an `INTEGER` array of length `20`. On return it contains default values. For further information, see Section 2.2.

`CNTL` is a `REAL` (`DOUBLE PRECISION` in the `D` version) array of length `4`. On return, it contains default values. For further information, see Section 2.2.

2.1.2 The conjugate gradient subroutine

The conjugate gradient subroutine is called as follows:

The single precision version

```
CALL MI31A(N, RHS, W, LDW, ICNTL, CNTL, IDO, INFO, RINFO, IPTR)
```

The double precision version

```
CALL MI31AD(N, RHS, W, LDW, ICNTL, CNTL, IDO, INFO, RINFO, IPTR)
```

`N` is an `INTEGER` variable of `INTENT(IN)` that must be set by the user to the number of columns n of the matrix $\mathbf{A}$. **Restriction**: `N`≥ 1.

`RHS` is a `REAL` (`DOUBLE PRECISION` in the `D` version) array of `INTENT(IN)` and length `N` that must be set by the user to hold the right-hand side $\mathbf{b}$ of the system; it is unchanged by the routine.

`W` is a `REAL` (`DOUBLE PRECISION` in the `D` version) array of dimension `(LDW,7)` such that

`W(1:N,1)` on input, if `ICNTL(2)` is nonzero, the array must be set by the user to store the initial estimate of the solution $\mathbf{u}^{(0)}$; on output the array contains the current estimate of the solution vector $\mathbf{u}^{(k)}$;

`W(1:N,2)` workspace that contains the conjugate direction vector $\mathbf{p}^{(k)}$;

`W(1:N,3)` workspace that contains an approximation for the residual vector $\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{u}^{(k)}$;

`W(1:N,4)` workspace;

`W(1:N,5)` workspace;

`W(1:N,6)` workspace;

`W(1:N,7)` workspace.

`LDW` is an `INTEGER` variable `INTENT(IN)` that must be set by the user to the leading dimension of the array `W`. **Restriction**: `LDW` $\geq$ `N`.

`ICNTL` is an `INTEGER` array of length `20` whose default values are set by `MI31I/ID`, see Section 2.1.1. It contains control parameters that can be set by the user, see Section 2.2. **Restriction**: `ICNTL(3) = 0,1,2,3, or 4`.

`CNTL` is a `REAL` (`DOUBLE PRECISION` in the `D` version) array of length `4` whose default values are set by `MI31I/ID`, see Section 2.1.1. It contains control parameters that can be set by the user, see Section 2.2.

`IDO` is an `INTEGER` variable that must be set by the user to a negative value to start a new solution. The user must not modify its value during the solution process. On return from `MI31A/AD`, it tells the user what to do before recalling `MI31A/AD`, or that the solution has been found. `MI31A/AD` will set it to the following values

0 the iteration has completed and either convergence has been achived, or `INFO(2)`< 0 and a error has been detected (see Section 2.4);

1 the user must supply the matrix-vector product

$$\texttt{W(:,4)} = \mathbf{A}\texttt{W(:,IPTR)}. \tag{2.1}$$

2 the user must supply the solution of the system:

$$\mathbf{M}\texttt{W(:,4)} = \texttt{W(:,IPTR)}. \tag{2.2}$$

INFO is an INTEGER array of length 10 that need not be set by the user. On return from MI31A/AD it contains data about the calculation and the error messages, see Section 2.3. The user must not modify its value.

RINFO is a REAL (DOUBLE PRECISION in the D version) array of length 40 that need not be set by the user. On return from MI31A/AD , it contains data about the calculation, see Section 2.3. The user must not modify its value.

IPTR is an INTEGER variable that need not be set by the user. On return from MI31A/AD , it indicates the column of the array W to be considered. The user must not modify its value.

2.2 Control parameters

ICNTL(1) controls whether the user wishes to use preconditioning. When ICNTL(1) is set to 0, no preconditioning is used. When it is set to a value other than 0, the user must supply the preconditioning. The default value is 0.

ICNTL(2) controls whether the user wishes to supply an initial estimate of the solution vector **u**. When this parameter is set to 0, the vector $\mathbf{u}^{(0)} = (0,0,\ldots,0)^T$ is used as the initial estimate. When it is set to a value other than 0, the user must supply an initial estimate in W(1:N,1) prior to the first call to MI31A/AD. It has default value 0.

ICNTL(3) controls which stopping criterion offered by the routine is to be used. The default value is 0. The possible choices are listed below (see Section 4 for the details).

0 the computed solution is accepted if

$$||\mathbf{b} - \mathbf{A}\mathbf{u}^{(k)}||_2 \le max(||\mathbf{b} - \mathbf{A}\mathbf{u}^{(0)}||_2 * \texttt{CNTL(1)}, \texttt{CNTL(2)}),$$

where $\mathbf{u}^{(0)}$ is the initial estimate of the solution.

1 a lower bound estimate of the error is given by using the Hestenes and Stiefel rule with delay d (equivalent to the Gauss quadrature formula) equal to ICNTL(4). The user must supply the value of the tolerance factor η (see (4.3) and (4.4) in Section 4) to be used in the stopping criterion in CNTL(1).

2 a lower bound estimate of the error is given by using the Gauss-Radau quadrature formula with ICNTL(4) internal points. The user must supply the value of the tolerance factor η (see (4.3) and (4.4) in Section 4) to be used in the stopping criterion in CNTL(1) and an estimate of the maximum eigenvalue of **A** in CNTL(3).

3 an upper bound estimate of the error is given by using the Gauss-Radau quadrature formula with ICNTL(4) internal points. The user must supply the value of the tolerance factor η (see (4.3) and (4.4) in Section 4) to be used in the stopping criterion in CNTL(1) and an estimate of the minimum eigenvalue of **A** in CNTL(4).

4 a lower and an upper estimate of the error is given by using the Gauss-Radau quadrature formula with ICNTL(4) internal points. The user must supply the value of the tolerance factor η (see (4.3) and (4.4) in Section 4) to be used in the stopping criterion in CNTL(1) and an estimate of the minimum and maximum eigenvalue of **A** in CNTL(3) and CNTL(4) respectively. In this case, the stopping criterion uses the value of the *upper bound* of the error.

ICNTL(4) holds the number of past iterates used to estimate the error in the stopping criteria selected by ICNTL(3) $>$ 0, that is the number of points in the quadrature formulae of Gauss (Hestenes and Stiefel rule) and Gauss-Radau, see the description in section 4.1. If ICNTL(3) $>$ 0, ICNTL(4) must be set by the user. Its default value is zero.

ICNTL(5) holds the maximum number of iterations allowed to be performed by the routine. Its default value is zero. If the value is not positive on a entry with IDO negative, MI31A/AD sets its value to N.

`ICNTL(6)` controls which formula the user wishes to use for computing the estimate of the energy norm of the solution.

The possible choices are:

0 (this is the default value) the estimator is based on a formula that does not require the use of the scalar product between $\mathbf{b}-\mathbf{A}\mathbf{u}^{(0)}$ and the current $\mathbf{u}^{(k)}$ (see (4.4) in Section 4.1);

1 the estimator is based on the formula (4.3) of Section 4.1.

The default value gives the cheapest choice, and, when $\mathbf{u}^{(0)}=0$ the estimates obtained by setting `ICNTL(6)` equal to 0 or to 1 are equal. Nevertheless, when $\mathbf{u}^{(0)}\neq 0$ and `ICNTL(6)` is equal to 0, the computation can present numerical instabilities when $||\mathbf{u}||^2_{\mathbf{A}}$ ≈machine precision. In this case the more costly formula given by choosing `ICNTL(6)` equal to 1, may be more stable.

`ICNTL(7)` is the unit number for error messages and has default value 6. If the user supplies a different unit numbers, messages are printed to the user supplied unit. If `ICNTL(7)` is negative, printing is suppressed.

`ICNTL(8)` is the unit number for warning messages and has default value 6. If the user supplies a different unit numbers, messages are printed to the user supplied unit. If `ICNTL(8)` is negative, printing is suppressed.

`CNTL(1)` and `CNTL(2)` are two convergence tolerances, see Section 4.1. The default values are `CNTL(1)`=$\sqrt{\varepsilon}$, where ε is the machine precision, and `CNTL(2)=0`.

- When `ICNTL(3)=0` the stopping criterion is

$$\textbf{stop when} \quad ||\mathbf{b}-\mathbf{A}\mathbf{u}^{(k)}||_2 \leq \max(||\mathbf{b}-\mathbf{A}\mathbf{u}^{(0)}||_2 * \texttt{CNTL(1)}, \texttt{CNTL(2)}), \tag{2.3}$$

where $\mathbf{u}^{(k)}$ is the current iterate solution and $\mathbf{u}^{(0)}$ is the initial estimate of the solution.

- When `ICNTL(3)`>0, the stopping criterion is

$$\textbf{stop when} \quad \textit{error\_est} \leq \textit{norm\_u\_est} * \texttt{CNTL(1)} \tag{2.4}$$

where *error_est* is the estimate provided by the selected criterion and *norm_u_est* is the estimate of the energy norm of the solution **u** selected by `ICNTL(6)`. In this case, `CNTL(1)` is the quantity η described in Section 4.1, and `CNTL(2)` must not be changed by the user between the calls of `MI31I/ID` and `MI31A/AD`.

`CNTL(3)` must be set by the user to an estimate of the minimum eigenvalue of the matrix **A** when the Gauss-Radau upper bound estimate of the error is required, that is `ICNTL(3)=3` or `ICNTL(3)=4`. Its default value is zero.

`CNTL(4)` must be set by the user to an estimate of the maximum eigenvalue of the matrix **A** when the Gauss-Radau lower bound estimate of the error is required, that is `ICNTL(3)=2` or `ICNTL(3)=4`. Its default value is zero.

2.3 Information returned to the user

`INFO(1)` returns the current iteration number.

`INFO(2)` is the output error flag. It informs the user about any error detected by the routine during the initialisation phase in the input parameters. A value of 0 means that no error is detected. See Section 2.4 for the list of the possible error messages.

`RINFO(1)` contains the value of *norm_u_est*, the estimate of the energy norm of the solution **u** selected by `ICNTL(6)`;

`RINFO(2)` contains the value of the Euclidean norm of the residual at the starting iteration, $||\mathbf{r}^{(0)}||$;

`RINFO(3)` contains the current value of the curvature $\mathbf{p}^{(k)T}\mathbf{A}\mathbf{p}^{(k)}/\mathbf{p}^{(k)T}\mathbf{p}^{(k)}$;

RINFO(4) contains the current value of the factor $\mathbf{z}^{(k)T)}\mathbf{r}^{(k)}$ (see Section 4);

RINFO(5) contains the previous value of the factor $\mathbf{z}^{(k-1)T}\mathbf{r}^{(k-1)}$ (see Section 4);

RINFO(6) contains the current value of α_{k-1} (see Section 4);

RINFO(7) contains the current value of β_k (see Section 4).

RINFO(8) contains the Euclidean norm of the residual at the current iteration, $||\mathbf{r}^{(k)}||$ (see Section 4);

RINFO(9) contains the square of the lower bound on the error estimate provided by the Hestenes and Stiefel formula;

RINFO(10) contains the square of the lower bound on the error estimate provided by the Gauss-Radau integration formula;

RINFO(11) contains the the square of upper bound on the error estimate provided by the Gauss-Radau integration formula.

The remaining values of INFO and RINFO are not used in this version.

2.4 Warning and error messages

If INFO(2) has zero value on output, the code did not detect any error in the initialization phase. Otherwise, a negative value indicates one of the following errors.

-1 incorrect value of N; this parameter must be a strictly positive integer value;

-2 incorrect value of LDW; LDW must be greater or equal to N;

-3 incorrect value in ICNTL(3); this parameter must be set to an integer value in the range $[0,4]$;

-4 incorrect value in ICNTL(4); this parameter must be set to a strictly positive integer value;

-5 incorrect value in CNTL(1); this parameter must be set to a strictly positive real number;

-6 incorrect value in CNTL(3); this parameter must be set to a strictly positive real number;

-7 incorrect value in CNTL(4); this parameter must be set to a strictly positive real number;

-8 incorrect value in CNTL(3) and CNTL(4); these two values must be set such that CNTL(3) is strictly less than CNTL(4).

-9 the matrix $\overline{\mathbf{A}}$ is singular: the curvature $\mathbf{p}^{(k)T}\overline{\mathbf{A}}\mathbf{p}^{(k)}/\mathbf{p}^{(k)T}\mathbf{p}^{(k)} = 0$;

-10 the preconditioning matrix M is singular: the factor $\mathbf{z}^{(k)T)}\mathbf{r}^{(k)} = 0$ but $\mathbf{r}^{(k)} \neq 0$.

A positive value of INFO(3) gives the following warning messages.

1 the matrix $\overline{\mathbf{A}}$ is indefinite; convergence is not guaranteed;

2 the matrix $\overline{\mathbf{A}}$ is ill-conditioned or close to be semidefinite positive; convergence is not guaranteed;

3 the preconditioning matrix M is not positive definite; convergence is not guaranteed;

4 ICNTL(5) was negative on a entry with IDO negative and MI31A/AD set ICNTL(5) equal to N.

3 GENERAL INFORMATION

Workspace: `W`

Other routines called directly: BLAS: `_DOT`, `_NRM2`, `_SCAL`, `_AXPY`.

Other modules used directly: None.

Input/output: Output is provided under the control of `ICNTL(7)` and `ICNTL(8)` for error and warning messages respectively.

Restrictions: `N`≥ 1, `LDW` $\geq$ `N`, `ICNTL(3) = 0,1,2,3,4`.

Portability: ISO Fortran 90.

4 METHOD

We implement the conjugate gradient method for solving the linear system of equations

$$\mathbf{Au} = \mathbf{b} \tag{4.1}$$

where $\mathbf{A} \in \mathbf{R}^{N\times N}$ is large, sparse, symmetric and positive definite and $\mathbf{b} \in \mathbf{R}^{N}$. In particular, we use a reverse communication approach, and we implement several stopping criteria. These stopping criteria are based on different backward error analysis choices and error estimators.

4.1 Error estimators and stopping criteria

4.1.1 Euclidean norm of the residual

When using an iterative method for solving the linear system (4.1), we can incorporate a stopping criterion based on the a posteriori component-wise or norm-wise backward error theory [2].

The common criterion of using the ratio of the residual to the right-hand side,

$$\omega = \frac{||\mathbf{r}(\mathbf{u}^{(k)})||_2}{||\mathbf{b}||_2}$$

implies that, if $\omega \neq \infty$, there is a vector $\delta\mathbf{b}$ with $||\delta\mathbf{b}||_2 \leq \omega||\mathbf{b}||_2$, such that

$$\mathbf{Au}^{(k)} = \mathbf{b} + \delta\mathbf{b}.$$

In practice, the conjugate gradient iterations are often stopped when the value of ω is less than or equal to the square root of machine precision. Nevertheless, the experiments in [1] give very good evidence that these stopping criteria based on the Euclidean norm of the residual $\mathbf{b} - \mathbf{Au}^{(k)}$ can be unsatisfactory and frequently misleading when the linear system (4.1) has originated from a finite-element approximation of a partial differential equation.

4.1.2 Energy norm of the residual

If we use the conjugate gradient method, it is quite natural to have a stopping criterion which takes advantage of the minimization property of this method. At each step k the conjugate gradient minimizes the energy norm of the error $\delta\mathbf{u} = \mathbf{u} - \mathbf{u}^{(k)}$ ($\mathbf{u}$ solution of (4.1)) on a Krylov space $\mathcal{K}_k$:

$$\min_{\mathbf{u}^{(k)} \in \mathcal{K}_k + \mathbf{u}_0} \delta\mathbf{u}^T \mathbf{A} \delta\mathbf{u}.$$

The space $\mathbf{R}^N$ with the norm

$$\|\mathbf{y}\|_{\mathbf{A}} = (\mathbf{y}^T\mathbf{A}\mathbf{y})^{1/2}$$

induces on its dual space the dual norm

$$\|\mathbf{f}\|_{\mathbf{A}^{-1}} = (\mathbf{f}^T\mathbf{A}^{-1}\mathbf{f})^{1/2}.$$

Therefore, a stopping criterion such as:

$$\text{IF } \|\mathbf{A}\mathbf{u}^{(k)} - \mathbf{b}\|_{\mathbf{A}^{-1}} \le \eta\|\mathbf{b}\|_{\mathbf{A}^{-1}} \text{ THEN STOP }, \tag{4.2}$$

with $\eta < 1$ an *a priori* threshold fixed by the user, will guarantee that a $\mathbf{u}^{(k)}$ which satisfies it, is the solution of the perturbed linear system:

$$\begin{array}{rcl} \mathbf{A}\mathbf{u}^{(k)} & = & \mathbf{b} - \mathbf{r}^{(k)}, \\ \|\mathbf{u} - \mathbf{u}^{(k)}\|_{\mathbf{A}} = \|\mathbf{r}^{(k)}\|_{\mathbf{A}^{-1}} & \le & \eta\|\mathbf{b}\|_{\mathbf{A}^{-1}} = \eta\|\mathbf{u}\|_{\mathbf{A}}. \end{array}$$

The choice of η will depend on the properties of the problem that we want to solve and, in practical cases, η can frequently be much larger than ε, the roundoff unit of the computer finite precision arithmetic. When (4.1) is a linear system obtained from the finite-element approximation of a partial-differential equation, a reasonable choice for η would be $\eta = h$ or $\eta = h^2$ where $h = \max_{T\in\mathcal{T}_h} \text{diameter}(T)$ with $\mathcal{T}_h$ a set of disjoint triangles $\{T\}$ which covers the domain Ω where the partial differential equation is defined (see [1] for a detailed analysis).

We implement two variants of the stopping criterion that estimate the value $\mathbf{e}_{\mathbf{A}}^{(k)} = \mathbf{r}^{(k)T}\mathbf{A}^{-1}\mathbf{r}^{(k)}$ at each step k:

- the variant of Gauss quadrature proposed by Hestenes and Stiefel that gives a lower bound τ_k for $\mathbf{e}_{\mathbf{A}}^{(k-d)}$;
- the Gauss-Radau quadrature variant that requires the estimate λ_{min} of the smallest eigenvalue of the preconditioned matrix $\mathbf{A}$ to give an upper bound Ξ_k for $\mathbf{e}_{\mathbf{A}}^{(k-d)}$, and the estimate λ_{max} of the biggest eigenvalue of the preconditioned matrix $\mathbf{A}$ to give a lower bound ξ_k for $\mathbf{e}_{\mathbf{A}}^{(k-d)}$.

In [1], the choice of the delay parameter d is discussed. If the convergence is reasonably fast (i.e. the preconditioner is effective) $d = 5$ is a successful compromise, and numerical experiments support this conclusion.

Finally, following [3], we implement two estimates of $\mathbf{b}^T\mathbf{A}^{-1}\mathbf{b}$. Therefore, (4.2) can be replaced by:

$$\text{IF } \varsigma_k \le \eta^2\left(\mathbf{b}^T\mathbf{u}^{(0)} + \mathbf{r}^{(0)T}\mathbf{u}^{(k)}\right) \text{ THEN STOP }, \tag{4.3}$$

or by

$$\text{IF } \varsigma_k \le \eta^2(\mathbf{r}^{(0)T}\mathbf{u}^{(0)} + \mathbf{b}^T\mathbf{u}^{(0)} + \sum_{j=1}^{k}\psi_j) \text{ THEN STOP }, \tag{4.4}$$

where ς_k is one of the estimates τ_k, ξ_k, or Ξ_k. Moreover, (4.3) needs only a scalar product at each step and, when $\mathbf{u}^{(0)} \ne 0$, may be more stable than (4.4).

If we introduce a preconditioner to speed up the convergence rate of the conjugate gradient algorithm, we do not need to update the previous technique for the evaluation of $\mathbf{e}_{\mathbf{A}}^{(k)}$ as can be seen in [1].

In Figure 1, we describe a variant of the preconditioned conjugate gradient algorithm which incorporates the proposed stopping criterion with a suitable choice of d.

Given an initial guess $\mathbf{u}^{(0)}$, compute $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{u}^{(0)}$, and solve $\mathbf{M}\mathbf{z}^{(0)} = \mathbf{r}^{(0)}$. Set $\mathbf{p}^{(0)} = \mathbf{z}^{(0)}$, $\beta_0 = 0$, $\alpha_{-1} = 1$, $\chi_1 = 1$, $\varsigma_0 = 1$, and $k = 0$:

```
while $\varsigma_k > \eta^2 norm\_u\_est$ do
    $k = k+1;$
    $\alpha_{k-1} = \frac{\mathbf{r}^{(k-1)T}\mathbf{z}^{(k-1)}}{\mathbf{p}^{(k-1)T}\mathbf{A}\mathbf{p}^{(k-1)}};\quad \psi_k = \frac{(\mathbf{r}^{(k-1)T}\mathbf{z}^{(k-1)})^2}{\mathbf{p}^{(k-1)T}\mathbf{A}\mathbf{p}^{(k-1)}};\quad \omega_k = \frac{1}{\alpha_{k-1}} + \frac{\beta_{k-1}}{\alpha_{k-2}};$
    if $k = 1$ then
        $\rho_1 = \omega_1;\quad \phi_1 = \omega_1 - \lambda_{min};\quad \Phi_1 = \omega_1 - \lambda_{max};$
    else
        $\chi_k = \frac{\chi_{k-1}\pi_{k-1}}{\rho_{k-1}};\quad \rho_k = \omega_k - \frac{\pi_{k-1}^2}{\rho_{k-1}};\quad \phi_k = \omega_k - \omega_{k-1}^u;\quad \Phi_k = \omega_k - \omega_{k-1}^l;$
    endif
    $\mathbf{u}^{(k)} = \mathbf{u}^{(k-1)} + \alpha_{k-1}\mathbf{p}^{(k-1)};$
    $\mathbf{r}^{(k)} = \mathbf{r}^{(k-1)} - \alpha_{k-1}\mathbf{A}\mathbf{p}^{(k-1)};$
    Solve $\mathbf{M}\mathbf{z}^{(k)} = \mathbf{r}^{(k)};$
    $\beta_k = \frac{\mathbf{r}^{(k)T}\mathbf{z}^{(k)}}{\mathbf{r}^{(k-1)T}\mathbf{z}^{(k-1)}};\quad \pi_k = \frac{\sqrt{\beta_k}}{\alpha_{k-1}};$
    $\mathbf{p}^k = \mathbf{z}^k + \beta_k\mathbf{p}^{(k-1)};$
    $\omega_k^u = \lambda_{min} + \frac{\pi_k^2}{\Phi_k};\quad \omega_k^l = \lambda_{max} + \frac{\pi_k^2}{\phi_k};\quad \psi_k^l = \frac{\chi_k^2\pi_k^2}{\rho_k(\omega_k^l\rho_k - \pi_k^2)};\quad \psi_k^u = \frac{\chi_k^2\pi_k^2}{\rho_k(\omega_k^u\rho_k - \pi_k^2)};$
    if $k > d$ then
        $\tau_k = \sum_{j=k-d+1}^{k} \psi_j;\quad \xi_k = \|\mathbf{r}^{(0)}\|_{M^{-1}}^2\psi_k^l + \tau_k;\quad \Xi_k = \|\mathbf{r}^{(0)}\|_{M^{-1}}^2\psi_k^u + \tau_k;$
        $\varsigma_k = \tau_k$   or$\{= \xi_k\}$   or$\{= \Xi_k\};$
        $norm\_u\_est = (\mathbf{r}^{(0)T}\mathbf{u}^{(0)} + \mathbf{b}^T\mathbf{u}^{(0)} + \sum_{j=1}^{k}\psi_j)$   or$\{= \mathbf{b}^T\mathbf{u}^{(0)} + \mathbf{r}^{(0)T}\mathbf{u}^{(k)}\}$
    endif
end while.
```

Figure 1: Preconditioned conjugate gradient algorithm (PCG)

References

[1] M. ARIOLI, A stopping criterion for the conjugate gradient algorithm in a finite element method framework. Numer. Math. electronic (2003).

[2] M. ARIOLI, I. S. DUFF, AND D. RUIZ, Stopping criteria for iterative solvers., SIAM J. Matrix Anal. and Appl., 13, (1992), pp. 138–144.

[3] M. ARIOLI AND G. MANZINI, MI31: a conjugate gradient algorithm implementation with energy-norm stopping criteria RAL-TR-2005-004

5 EXAMPLE OF USE

Suppose we wish to solve the problem $\mathbf{Ax} = \mathbf{b}$, where

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & & & & & & & & \\ -1 & 2 & -1 & & & & & & & \\ & -1 & 2 & -1 & & & & & & \\ & & -1 & 2 & -1 & & & & & \\ & & & -1 & 2 & -1 & & & & \\ & & & & -1 & 2 & -1 & & & \\ & & & & & -1 & 2 & -1 & & \\ & & & & & & -1 & 2 & -1 & \\ & & & & & & & -1 & 2 & -1 \\ & & & & & & & & -1 & 2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}. \tag{5.1}$$

The following program uses the conjugate gradient method and a diagonal Jacobi preconditioner. The iterations are stopped by using the the lower bound on the error estimate provided by the Gauss quadrature formula.

```
program main
  use hsl_mi31_double
  implicit none

  integer, parameter :: n    = 10, ldw  = n
  double precision, parameter :: zero = 0.0d0, one  = 1.0d0, two  = 2.0d0
  double precision, dimension(n)   :: rhs
  double precision, dimension(n,7) :: w
  integer,          dimension(10)  :: info
  double precision, dimension(40)  :: rinfo
  integer,          dimension(20)  :: icntl
  double precision, dimension(4)   :: cntl
  integer :: Ido, iptr, i

  rhs(1:n) = one/(n*n) !! right-hand-side
  w = 0.d0    !! set w
  w(1:n,1) = one !! initial solution
  !! set pars
  info  = 0
  rinfo = 0.d0

  call mi31id( icntl, cntl )

  !! set icntl
  icntl(1) = 1    !! external preconditioner
  icntl(2) = 1    !! user initial solution
  icntl(3) = 1    !! uses Hestenes-Stiefel
  icntl(4) = 3    !! uses 3 quadrature points
  !! set cntl
  cntl(1) = 1.d-6 !! convergence tolerance

  Ido = -1     !! start iterations
  do while ( Ido .ne. 0 )

     call mi31ad( n, rhs, w, ldw, icntl, cntl, Ido, info, rinfo, iptr )

     select case ( Ido )

     case (1) !! Matrix-Vector product
        w(1,4) = two*w(1,iptr) - w(2,iptr)
        do i=2,n-1
           w(i,4) = -w(i-1,iptr) + two*w(i,iptr) - w(i+1,iptr)
        end do
        w(n,4) = - w(n-1,iptr) + two*w(n,iptr)
     case (2) !! Jacobi preconditioner
        do i=1,n
           w(i,4) = w(i,iptr)/two
        end do

     end select

  end do

  if (info(2) .ge. 0) then
     write(*,'(a,i5)') 'error code  = ',info(2)
     !! print the final solution
     write(*,'(a,i5)') 'number of iterations  = ',info(1)
     write(*,'(a)') '    i       X(i)'
     write(*,'(i5,1x,1pe14.7)') (i,w(i,1), i=1,n)
  end if
end program main
```

This produces the following output:

```
error code  =     0
number of iterations  =     8
    i       X(i)
    1  5.0000000E-02
    2  9.0000000E-02
    3  1.2000000E-01
    4  1.4000000E-01
    5  1.5000000E-01
    6  1.5000000E-01
    7  1.4000000E-01
    8  1.2000000E-01
    9  9.0000000E-02
   10  5.0000000E-02
```