Science and Technology Facilities Council

HSL_MI27

PACKAGE SPECIFICATION HSL

1 SUMMARY

This package uses the **projected preconditioned conjugate gradient method** to solve $(n+m)\times(n+m)$ **saddle-point systems** of the form

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}^{\mathbf{T}} \\ \mathbf{B} & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix},$$

where $\mathbf{A}$ is an $n\times n$ real and symmetric matrix, $\mathbf{C}$ is an $m\times m$ real, symmetric and positive semi-definite (possibly zero) matrix, and $m\leq n$. A preconditioner of the form

$$\mathbf{P} = \begin{bmatrix} \mathbf{G} & \mathbf{B}^{\mathbf{T}} \\ \mathbf{B} & -\mathbf{C} \end{bmatrix}$$

must be available where $\mathbf{G}$ is a real and symmetric matrix. The following assumptions are assumed to hold:

- if $\mathbf{C}$ is positive definite, both $\mathbf{A}+\mathbf{B}^{\mathbf{T}}\mathbf{C}^{-1}\mathbf{B}$ and $\mathbf{G}+\mathbf{B}^{\mathbf{T}}\mathbf{C}^{-1}\mathbf{B}$ are positive definite;
- if $\mathbf{C}=0$ and the columns of the $n\times(n-m)$ matrix $\mathbf{Z}$ span the nullspace of $\mathbf{B}$, both $\mathbf{Z}^{\mathbf{T}}\mathbf{A}\mathbf{Z}$ and $\mathbf{Z}^{\mathbf{T}}\mathbf{G}\mathbf{Z}$ are positive definite;
- if $\mathbf{C}=\mathbf{E}\mathbf{D}\mathbf{E}^{\mathbf{T}}$, where $\mathbf{D}$ is a $p\times p$ nonsingular matrix with $0<p<m$, the columns of the $m\times(m-p)$ matrix $\mathbf{F}$ span the nullspace of $\mathbf{C}$ and the columns of the $n\times(n-m+p)$ matrix $\mathbf{Z}$ span the nullspace of $\mathbf{F}^{\mathbf{T}}\mathbf{B}$, then both $\mathbf{Z}^{\mathbf{T}}\left(\mathbf{A}+\mathbf{B}^{\mathbf{T}}\mathbf{E}\mathbf{D}^{-1}\mathbf{E}^{\mathbf{T}}\mathbf{B}\right)\mathbf{Z}$ and $\mathbf{Z}^{\mathbf{T}}\left(\mathbf{G}+\mathbf{B}^{\mathbf{T}}\mathbf{E}\mathbf{D}^{-1}\mathbf{E}^{\mathbf{T}}\mathbf{B}\right)\mathbf{Z}$ are positive definite.

If these assumptions do not hold, then negative curvature may occur and, consequently, the method terminates with an error.

The projected preconditioned conjugate gradient method iteratively finds the vector $\mathbf{x}$ and then, once $\mathbf{x}$ has been computed to a high enough level of accuracy, the vector $\mathbf{y}$ is computed by performing one additional solve with the preconditioner $\mathbf{P}$. Reverse communication is used for preconditioning and matrix-vector products of the form $\mathbf{As}$, $\mathbf{Bs}$, $\mathbf{B^Ts}$ and $\mathbf{Cs}$. HSL_MI13 may be used to efficiently form suitable preconditioners and carry out the required preconditioning solves; HSL_MC65 may be used to form the required matrix-vector products. HSL_MI13 and HSL_MC65 are both available as part of the current version of HSL.

Reference

[1] H. S. Dollar, N. I. M. Gould, W. H. A. Schilders and A. J. Wathen, *Implicit-factorization preconditioning and iterative solvers for regularized saddle-point systems*, SIAM Journal on Matrix Analysis and Applications, 28(2006), pp. 170–189.

ATTRIBUTES — Version: 1.1.0 (01 March 2023) **Types:** Real (single, double). **Uses:** The BLAS subroutines _AXPY, _COPY, _DOT, _NRM2, _SCAL. **Date:** January 2011. **Origin:** H. S. Thorne, Rutherford Appleton Laboratory. **Language:** Fortran 95, plus allocatable dummy arguments and allocatable components of derived types. **Remark:** The development of HSL_MI27 was funded by EPSRC grant EP/E053351/1.

2 HOW TO USE THE PACKAGE

2.1 Calling sequences

Access to the package requires a USE statement:

Single precision version

```
USE HSL_MI27_single
```

Double precision version

```
USE HSL_MI27_double
```

In MI27_single, all reals are default reals. In MI27_double, all reals are double precision reals.

If it is required to use both modules at the same time, the derived types (Section 2.2) must be renamed in one of the USE statements.

The following procedures are available to the user:

MI27_setup takes the sizes of the matrices **A** and **C**, and generates the data structures that are required by the projected preconditioned conjugate gradient method.

MI27_ppcg uses the projected preconditioned conjugate gradient method to solve

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}^{\mathbf{T}} \\ \mathbf{B} & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix}.$$

MI27_ppcg uses reverse communication for preconditioning operations and matrix-vector products.

MI27_finalize should be called after all other calls to HSL_MI27 are completed for a particular size of **A** and **C**, and choice of controls. MI27_finalize deallocates components of the derived types.

2.2 The derived data types

For each problem, the user must employ the derived types defined by the module to declare scalars of the types MI27_keep, MI27_control and MI27_info. The following pseudocode illustrates this.

```
use HSL_MI27_double
...
type (MI27_keep) :: keep
type (MI27_control) :: control
type (MI27_info) :: info
...
```

The components of MI27_keep are private and are used to pass data between the subroutines of the package. The components of the other derived types are explained in Sections 2.3.4 and 2.3.5.

2.3 Argument lists and calling sequences

2.3.1 The PPCG setup phase

To set up the data structures for a particular size of **A**, size of **C**, and set of control paremeters, a call of the following form must be made:

```
call MI27_setup(n,m,control,keep,info)
```

n a scalar of INTENT(IN) argument of type default INTEGER that must hold the number of rows in **A**.
Restriction: n $\geq$ 1.

m a scalar of INTENT(IN) argument of type default INTEGER that must hold the number of rows in **B**.
Restriction: 1 $\leq$ m $\leq$ n.

keep is a scalar INTENT(OUT) argument of type MI27_keep. It is used to hold data about the preconditioner and must be passed unchanged to the other subroutines.

control a scalar INTENT(IN) argument of type MI27_control (see Section 2.3.4).

info a scalar INTENT(OUT) argument of type MI27_info (see Section 2.3.5). On successful exit, info%flag is set to 0.

2.3.2 To solve the saddle-point problem

The projected preconditioned conjugate gradient method may be applied to compute the vectors **x** and **y** in the solution vector of the saddle-point problem

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}^{\mathbf{T}} \\ \mathbf{B} & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix},$$

by making a series of calls as follows.

```
call MI27_ppcg(action,w,ldw,locq,locs,locu,locv,control,keep,info)
```

action is a scalar INTENT(INOUT) argument of type INTEGER. Prior to the first call to MI27_ppcg, action must be set by the user to 0. On each exit, action indicates the action required by the user. Possible values of action and the action required are as follows:

-1 An error has occured and the user must terminate the computation (see info%flag).

1 control%convergence=.true. and convergence has been achieved. The computed vector **x** is held in the first n entries of column 3 of the array w. The user may call MI27_ppcg with action=3 and the remaining arguments unchanged to compute the vector **y**.

2 control%convergence=.false. and the current estimate **x** is held in the first n entries of column 3 of the array w. If the user does not wish to test for convergence or determines that convergence has not been achieved, he or she must recall MI27_ppcg with action=2 and the remaining arguments unchanged. If the user requires the vector **y** corresponding to the current vector **x** for their convergence test or if the user determines that **x** has converged and requires the corresponding vector **y**, the user may recall MI27_ppcg with action=3 and the remaining arguments unchanged to compute the vector **y**.

3 The vector **y** corresponding to the current estimate **x** has been computed and is held in the first m entries of column 4 of the array w; the current estimate **x** is held in the first n entries of column 3 of the array w. If convergence has not been achieved, the user must recall MI27_ppcg with action=2 and the remaining arguments unchanged.

4 The user must perform the matrix-vector product

$$\mathbf{q} := \mathbf{A}\mathbf{s}$$

and recall MI27_ppcg. The vectors **q** and **s** are held in the first n entries of the columns locq and locs of array w, respectively.

5 The user must perform the matrix-vector product

$$\mathbf{q} := \mathbf{B}\mathbf{s}$$

and recall MI27_ppcg. The vector **q** is held in the first m entries of the column locq of array w and vector **s** is held in the first n entries of the column locs of array w.

6 The user must perform the matrix-vector product

$$\mathbf{q} := \mathbf{B}^{\mathbf{T}}\mathbf{s}$$

and recall MI27_ppcg. The vector **q** is held in the first n entries of the column locq of array w and vector **s** is held in the first m entries of the column locs of array w.

7 The user must perform the matrix-vector product

$$\mathbf{q} := \mathbf{C}\mathbf{s}$$

and recall MI27_ppcg. The vectors **q** and **s** are held in the first m entries of the columns locs and locu of array w, respectively.

8 The user must perform the preconditioning operation

$$\text{Solve} \begin{bmatrix} \mathbf{G} & \mathbf{B}^{\mathbf{T}} \\ \mathbf{B} & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \mathbf{s} \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}$$

and recall MI27_ppcg. The vectors **q** and **u** are held in the first n entries of the columns locq and locu of array w, respectively. The vectors **s** and **v** are held in the first m entries of the columns locs and locv of array w, respectively.

w is a rank-2 array INTENT(INOUT) argument of type REAL with extents ldw and 12. Prior to the first call, the first n entries of column 1 must be set to the vector **c** and the first m entries of column 2 must be set to the vector **d**. If control%initial_x=.true., the first n entries of column 3 must be set to the initial estimate of **x**. On exit with action=1 or action=2, the current estimate of the solution **x** is held in the first n entries of column 3. On exit with action=3, the current estimate of the solution **x** is held in the first n entries of column 3 and the current estimate of the solution **y** is held in the first m entries of column 4. If action>3, the user is required to form **q** and possibly **s** in columns locq and locs, respectively, of w. The remaining columns of w must not be altered by the user between calls to MI27_ppcg.

ldw is a scalar INTENT(IN) argument of type default INTEGER that must be set by the user to the first dimension of the array w. **Restriction:** ldw≥n.

locq, locs, locu and locv are scalar INTENT(INOUT) arguments of type default INTEGER. On exit with action>1, they indicate the columns of w that contain the vectors **q**, **s**, **u** and **v** (see argument action). These arguments must not be altered by the user between calls to mi27_ppcg.

control a scalar INTENT(IN) argument of type MI27_control (see Section 2.3.4).

keep is a scalar INTENT(INOUT) argument of type MI27_keep that must be passed unchanged by the user.

info a scalar INTENT(OUT) argument of type MI27_info. On exit, it contains information (see Section 2.3.5).

2.3.3 The finalization subroutine

A call of the following form should be made after all other calls are complete for a particular problem size and choice of controls (including after an error return that does not allow the computation to continue) to deallocate components of the derived data types.

```
call MI27_finalize(control, keep, info)
```

keep is a scalar INTENT(INOUT) argument of type MI27_keep that must be passed unchanged. On exit, allocatable components will have been deallocated.

control a scalar INTENT(IN) argument of type MI27_control (see Section 2.3.4).

info a scalar INTENT(OUT) argument of type MI27_info. On exit, it contains information (see Section 2.3.5).

2.3.4 The derived data type MI27_control for holding control parameters

The derived data type MI27_control is used to hold controlling data. The components, which are automatically given default values in the definition of the type, are as follows.

Controls used by MI27_setup (in alphabetical order)

absolute_tol is a scalar of type REAL that defines the absolute convergence tolerance, see the control convergence. It has default value zero. If convergence=.false., absolute_tol is not used.

c_zero is a scalar of type default LOGICAL. If $\mathbf{C} = 0$, then setting c_zero=.true. will allow a simplified version of the algorithm to be used and the user will not be asked to form matrix-vector products of the form $\mathbf{q} = \mathbf{C}\mathbf{s}$. The default value is .false..

convergence is a scalar of type default LOGICAL that controls whether the convergence test offered by HSL_MI27 is to be used. It has default .true. and in this case a solution is accepted if $\sigma^{(i)}$ is less than or equal to max($\sigma^{(0)}$*relative_tol,absolute_tol), where $\sigma^{(i)}$ is defined in Section 4. If the user does not want to use this test, convergence should be equal to .false.. In this case, the user may test for convergence when action=1 is returned. **Note:** the projected preconditioned conjugate gradient method iteratively solves for $\mathbf{x}$: if the user wishes to compute the corresponding vector $\mathbf{y}$ for use in their convergence test, he or she may call MI27_ppcg with action=3 and all other aurguments unchanged prior to carrying out the convergence test.

curvature_tol is a scalar of type REAL that determines the smallest allowable curvature (see Section 4). It has default value u, where u = EPSILON(relative_tol). Values of curvature_tol that are less than or equal to zero are treated as if they were the default u.

initial_x is a scalar of type LOGICAL that controls whether the user wishes to supply an initial estimate of the solution vector $\mathbf{x}$. It has default value .false. and in this case $\mathbf{x} = (0,0,\ldots,0)^T$ is used as the initial estimate. If the user wishes to supply an initial estimate, initial_x must be equal to .true. and the initial estimate placed in the first n entries of column 3 of array w on the first call of MI27_ppcg.

max_iterations is a scalar of type INTEGER that determines the maximum number of iterations allowed. It has default -1 and in this case the maximum number of iterations allowed is $n+m$. Otherwise, max_iterations should be set to be the maximum number of iterations the user wishes to allow. Values of max_iterations that are less than or equal to zero are treated as if they were the default -1.

relative_tol is a scalar of type REAL that defines the relative convergence tolerance. It has default value 10^{-6}. If convergence=.false., relative_tol is not used.

update_tol is a scalar of type REAL that defines the residual update tolerance. It has default value 10^{-6}. If $||\mathbf{g}||_2 \leq$update_tol$*||\mathbf{v}||_2$, where $\mathbf{g}$ and $\mathbf{v}$ are defined in Section 4, the residual update strategy will be applied. If update_tol is negative, the residual update strategy will not be used.

Printing controls

error is a scalar of type default INTEGER that holds the unit number for the printing of error and warning messages. Printing is suppressed if error < 0. The default is 6.

print is a scalar of type default INTEGER that holds the unit number for diagnostic printing. Printing is suppressed if print < 0. The default is 6.

print_level is a scalar of type default INTEGER that controls the amount of printing. Possible values are:

0: no printing

1: printing of errors and warnings only

2: as 1 plus basic diagnostic printing

3: as 2 plus some more detailed diagnostic printing

The default is 1. Values of print_level that are less than zero are treated as if they were zero. Values of print_level that are greater than 3 are treated as if they were 3.

2.3.5 The derived data type MI27_info for holding information

The components of the derived data type MI27_info are used to provide information about the progress of the algorithm. The components of MI27_info are:

flag is a scalar of type default INTEGER that is used as a error and warning flag. See Section 2.4 for details.

iterations is a scalar of type default INTEGER that, after a call to MI27_ppcg, contains the number of iterations performed.

stat is a scalar of type default INTEGER that is used to hold the Fortran stat parameter.

2.4 Warning and error messages

A successful return from a subroutine in the package is indicated by info%flag having the value zero. A negative (respectively, positive) value is associated with an error (respectively, warning) message that by default will be output on unit control%error. Possible non-zero values are listed below.

2.4.1 Errors and warnings associated with MI27_setup

-1 n<1.

-2 m<1 or m>n.

-3 Allocation error.

-4 Deallocation error.

$+1$ The user-supplied convergence tolerance defined by the control parameter relative_tol lies outside the interval $(\mathrm{u}, 1.0)$, where $\mathrm{u} =$ EPSILON(relative_tol), and the control parameter convergence=.true.. The control parameter relative_tol is replaced by 10^{-6} in the convergence test.

2.4.2 Errors and warnings associated with MI27_ppcg

-5 ldw<n.

-6 The curvature encountered is too small and the algorithm has broken down (see Section 4).

-7 The maximum number of iterations determined by the control parameter max_iterations has been exceeded.

2.4.3 Errors and warnings associated with MI27_finalize

-4 Deallocation error.

3 GENERAL INFORMATION

Workspace: Provided automatically by the module.

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

Input/output: Output is provided under the control of control%print_level. In the event of an error or warning, diagnostic messages are printed. The output units for these messages are controlled by control%print, and control%error (see Section 2.3.4).

Restrictions: 1≤m≤n and ldw≥n

Portability: Fortran 95, plus allocatable dummy arguments and allocatable components of derived types.

4 METHOD

The projected preconditioned conjugate gradient method for solving systems of the form

$$\begin{bmatrix} \mathbf{A} & \mathbf{B^T} \\ \mathbf{B} & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix}, \tag{4.1}$$

is described in detail in [1]. The method proceeds by iteratively finding the vector $\mathbf{x}$ in (4.1). The corresponding vector $\mathbf{y}$ can be determined by carrying out one additional solve with the preconditioner. The algorithm used by MI27_ppcg takes the following form:

Find $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ such that $\mathbf{B}\hat{\mathbf{x}} - \mathbf{C}\hat{\mathbf{y}} = \mathbf{d} - \mathbf{B}\mathbf{x}^{(0)}$ by solving (4.2)
Set $\mathbf{x}^{(0)} \leftarrow \mathbf{x}^{(0)} + \hat{\mathbf{x}}$, $\mathbf{r}^{(0)} = \mathbf{A}\mathbf{x}^{(0)} + \mathbf{B}^\mathbf{T}\hat{\mathbf{y}} - \mathbf{c}$, $\mathbf{a}^{(0)} = \mathbf{0}$ and $\mathbf{w}^{(0)} = \mathbf{0}$
Solve

$$\begin{bmatrix} \mathbf{G} & \mathbf{B^T} \\ \mathbf{B} & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{g}^{(0)} \\ \mathbf{v}^{(0)} \end{bmatrix} = \begin{bmatrix} \mathbf{r}^{(0)} \\ \mathbf{w}^{(0)} \end{bmatrix}.$$

if $(||\mathbf{g}^{(0)}||_2 \leq$control%update_tol$||\mathbf{v}^{(0)}||_2)$ **then**
  Carry out residual update, see Section 4.2.
end if
Set $\mathbf{t}^{(0)} = \mathbf{v}^{(0)} + \mathbf{a}^{(0)}$, $\mathbf{p}^{(0)} = -\mathbf{g}^{(0)}$, $\mathbf{h}^{(0)} = -\mathbf{t}^{(0)}$, $\mathbf{q}^{(0)} = \mathbf{A}\mathbf{p}^{(0)}$ and $\mathbf{l}^{(0)} = \mathbf{C}\mathbf{h}^{(0)}$
Set $\sigma^{(0)} = (\mathbf{r}^{(0)})^T\mathbf{g}^{(0)} + (\mathbf{w}^{(0)})^T\mathbf{t}^{(0)}$ and $\gamma^{(0)} = (\mathbf{p}^{(0)})^T\mathbf{q}^{(0)} + (\mathbf{h}^{(0)})^T\mathbf{l}^{(0)}$
if convergence test is satisfied **then**
  Return with $\mathbf{x} = \mathbf{x}^{(0)}$
end if
if $\sigma^{(0)} < 0$ **or** $\gamma^{(0)} <$curvature_tol **then**
  Curvature too small and method has broken down. Set info%flag and return with action=-1.
end if
for $i = 1, 2, \ldots,$max_iterations **do**
  $\alpha^{(i)} = \sigma^{(i-1)}/\gamma^{(i-1)}$
  $\mathbf{x}^{(i)} = \mathbf{x}^{(i-1)} + \alpha^{(i)}\mathbf{p}^{(i-1)}$, $\mathbf{r}^{(i)} = \mathbf{r}^{(i-1)} + \alpha^{(i)}\mathbf{q}^{(i-1)}$, $\mathbf{a}^{(i)} = \mathbf{a}^{(i-1)} + \alpha^{(i)}\mathbf{h}^{(i-1)}$ and $\mathbf{w}^{(i)} = \mathbf{w}^{(i-1)} + \alpha^{(i)}\mathbf{l}^{(i-1)}$
  Solve

$$\begin{bmatrix} \mathbf{G} & \mathbf{B^T} \\ \mathbf{B} & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{g}^{(i)} \\ \mathbf{v}^{(i)} \end{bmatrix} = \begin{bmatrix} \mathbf{r}^{(i)} \\ \mathbf{w}^{(i)} \end{bmatrix}.$$

  if $(||\mathbf{g}^{(i)}||_2 \leq$control%update_tol$||\mathbf{v}^{(i)}||_2)$ **then**
    Carry out residual update, see Section 4.2.
  end if

Set $\mathbf{t}^{(i)} = \mathbf{a}^{(i)} + \mathbf{v}^{(i)}$
Set $\sigma^{(i)} = (\mathbf{r}^{(i)})^T\mathbf{g}^{(i)} + (\mathbf{w}^{(i)})^T\mathbf{t}^{(i)}$ and $\beta^{(i)} = \sigma^{(i)}/\sigma^{(i-1)}$
Set $\mathbf{p}^{(i)} = -\mathbf{g}^{(i)} + \beta^{(i)}\mathbf{p}^{(i-1)}$, $\mathbf{h}^{(i)} = -\mathbf{t}^{(i)} + \beta^{(i)}\mathbf{h}^{(i-1)}$, $\mathbf{q}^{(i)} = \mathbf{A}\mathbf{p}^{(i)}$ and $\mathbf{l}^{(i)} = \mathbf{C}\mathbf{h}^{(i)}$
Set $\gamma^{(i)} = (\mathbf{p}^{(i)})^T\mathbf{q}^{(i)} + (\mathbf{h}^{(i)})^T\mathbf{l}^{(i)}$
if convergence test is satisfied **then**
Return with $\mathbf{x} = \mathbf{x}^{(\mathbf{i})}$
end if
if $\sigma^{(i)} < 0$ **or** $\gamma^{(0)} <$`curvature_tol` **then**
Curvature too small and method has broken down. Set `info%flag` and return with `action=-1`.
end if
if maximum number of iterations have been performed **then**
Set `info%flag` and return with `action=-1`.
end if
end for

4.1 Finding $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ that satisfy $\mathbf{B}\hat{\mathbf{x}} - \mathbf{C}\hat{\mathbf{y}} = \mathbf{d} - \mathbf{B}\mathbf{x}^{(0)}$

The vectors $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ that satisfy $\mathbf{B}\hat{\mathbf{x}} - \mathbf{C}\hat{\mathbf{y}} = \mathbf{d} - \mathbf{B}\mathbf{x}^{(0)}$ are found by solving

$$\begin{bmatrix} \mathbf{G} & \mathbf{B^T} \\ \mathbf{B} & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}} \\ \hat{\mathbf{y}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{d} - \mathbf{B}\mathbf{x}^{(0)} \end{bmatrix}. \tag{4.2}$$

4.2 Residual update

The residual update modifies the vectors $\mathbf{a}^{(i)}$, $\mathbf{g}^{(i)}$, $\mathbf{r}^{(i)}$, $\mathbf{v}^{(i)}$ and $\mathbf{w}^{(i)}$ by computing

$$\mathbf{r}^{(i)} \leftarrow \mathbf{r}^{(i)} - \mathbf{B^T}\mathbf{v}^{(i)}, \mathbf{a}^{(i)} \leftarrow \mathbf{a}^{(i)} + \mathbf{v}^{(i)}, \mathbf{w}^{(i)} = \mathbf{C}\mathbf{a}^{(i)},$$

and then solving

$$\begin{bmatrix} \mathbf{G} & \mathbf{B^T} \\ \mathbf{B} & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{g}^{(i)} \\ \mathbf{v}^{(i)} \end{bmatrix} = \begin{bmatrix} \mathbf{r}^{(i)} \\ \mathbf{w}^{(i)} \end{bmatrix}.$$

4.3 Recovering $\mathbf{y}^{(i)}$ from $\mathbf{x}^{(i)}$,

Given an approximation $\mathbf{x}^{(i)}$, the corresponding vector $\mathbf{y}^{(i)}$ is found by solving

$$\begin{bmatrix} \mathbf{G} & \mathbf{B^T} \\ \mathbf{B} & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}} \\ \mathbf{y}^{(i)} \end{bmatrix} = \begin{bmatrix} \mathbf{c} - \mathbf{A}\mathbf{x}^{(i)} \\ \mathbf{d} - \mathbf{B}\mathbf{x}^{(i)} \end{bmatrix}.$$

Reference:

[1] H. S. Dollar, N. I. M. Gould, W. H. A. Schilders and A. J. Wathen, *Implicit-factorization preconditioning and iterative solvers for regularized saddle-point systems*, SIAM Journal on Matrix Analysis and Applications, 28(2006), pp. 170–189.

5 EXAMPLE OF USE

Suppose we wish to solve the linear system

$$\begin{bmatrix} \mathbf{A} & \mathbf{B^T} \\ \mathbf{B} & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

with

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1 & 1 & 2 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 2 \end{bmatrix}, \mathbf{c} = \begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix} \quad \text{and} \quad \mathbf{d} = \begin{bmatrix} 2 \end{bmatrix}.$$

A preconditioner of the form

$$\mathbf{P} = \begin{bmatrix} \mathbf{G} & \mathbf{B^T} \\ \mathbf{B} & -\mathbf{C} \end{bmatrix},$$

where

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

is applied. We may use the following code:

```
PROGRAM MAIN
        USE hsl_mi27_double

        ! Precision
        INTEGER, PARAMETER :: wp = kind(1.0D0)

        TYPE (mi27_keep) :: keep
        TYPE (mi27_control) :: control
        TYPE (mi27_info) :: info
        INTEGER :: m,n,nm ! size of A and C
        INTEGER :: i, problem ! local dummy variable
        INTEGER :: ldw,locq,locs,locu,locv,stat,action
        REAL (wp),allocatable, dimension(:,:) :: w
        REAL (wp),allocatable, dimension(:) :: r

        ! Set problem size
        n = 3; m = 1; nm = n + m

        ! Allocate arrays w and r
        ldw = n
        allocate(w(ldw,12),r(nm),stat=stat)
        IF (stat.ne.0) THEN
                write(6,'(a)') 'Allocation error'
                stop
        END IF

        ! Set rhs
        w(1,1)= 2.0_wp; w(2,1)= 3.0_wp; w(3,1)= 5.0_wp; w(1,2)= 2.0_wp

        ! Set-up data structures
        call mi27_setup(n,m,control,keep,info)
```

```
        ! Carry out iteration
        action = 0
        do while (action.ne.3 .and. action.ne.-1 )
                select case (action)
                case (1)
                        ! Iteration has converged: find y
                        action = 3
                case (4)
                        ! Form q = A*s
                        w(1,locq) = w(1,locs); w(2,locq) = 2.0*w(2,locs)
                        w(3,locq) = 3.0*w(3,locs)
                case (5)
                        ! Form q = B*s
                        w(1,locq) = w(1,locs) + w(2,locs) + 2.0*w(3,locs)
                case (6)
                        ! Form q = B^T*s
                        w(1,locq) = w(1,locs); w(2,locq) = w(1,locs)
                        w(3,locq) = 2.0*w(1,locs)
                case (7)
                        ! Form q = C*s
                        w(1,locq) = 2.0*w(1,locs)
                case (8)
                        ! Solve P [q;s] = [u;v]
                        w(1,locs) = w(1,locu)
                        w(2,locq) = (w(2,locu)-w(1,locs))
                        w(3,locq) = (w(3,locu)-2.0*w(1,locs))
                        w(1,locq) = w(1,locv)-w(2,locq)-2.0*w(3,locq)&
                                +2.0*w(1,locs)
                end select
                call mi27_ppcg(action,w,ldw,locq,locs,locu,locv,control,keep,&
                                info)
        end do

        write(6,'(a4)') 'x = '; write(6,'(5f8.3)') w(1:n,3)
        write(6,'(a4)') 'y = '; write(6,'(5f8.3)') w(1:m,4)

        call mi27_finalize(control,keep,info)

        deallocate(w,r,stat=stat)
        IF (stat.ne.0) THEN
                write(6,'(a)') 'Deallocation error'
                stop
        END IF

END PROGRAM MAIN
```

This produces the following output:

```
x =
   1.000   1.000   1.000
y =
   1.000
residual =
   0.000  -0.000  -0.000   0.000
iterations =  3
```