Exercises on GMRES and MINRES
Advanced Numerical Analysis (2019–2020).

1 Nonsymmetric linear systems

The GMRES method is implemented in MATLAB and can be used with the following syntax

```
[x, flag, relres, iter, resvec] = gmres(A, b, restart, tol, maxit, M1, M2, x0)
```

For the meaning of the **input** parameters, see the description of the CG method in the previous set of exercises (homework 2) or the MATLAB documentation obtained invoking the online help on the `gmres` function. The GMRES method needs an additional parameter, `restart` which indicates the number of times that iteration can start (the maximum number of restarts plus one). In this case the parameter `maxit` denotes the maximum number of **outer** iterations i.e. the number of **restarts** decreased by one.

Regarding the preconditioner, if the ILU factorization ($A \approx LU$) is used, then `M1` is L while `M2` is U.

The meaning of the **output** parameters are the same as those of the PCG method with one exception: the output parameter `iter` is a vector with 2 components: `iter(1)` is the number of times the iteration has been started (number of restarts + 1) while `iter(2)` counts the number of iterations after the last restart. The actual number of iterations employed must be computed as

$$(\texttt{iter}(1) - 1) * \texttt{restart} + \texttt{iter}(2)$$

Note. MATLAB `gmres` implements left preconditioning. Hence vector `resvec` stores the norms of the absolute preconditioned residuals, while `relres` is $\frac{\|M^{-1}\mathbf{r}_k\|}{\|M^{-1}\mathbf{b}\|}$ and consequently the exit test performed is $\frac{\|M^{-1}\mathbf{r}_k\|}{\|M^{-1}\mathbf{b}\|} < \texttt{TOL}$.

1.1 ILU preconditioner

The ILU preconditioner can be computed using the `ilu` MATLAB function with the following syntax:

```
[L,U] = ilu(A, setup);
```

where the `setup` structure has (among the others) the following fields

```
setup.type = 'crout';
setup.droptol = 0.01;
```

If the setup parameter is not specified the command

```
[L,U] = ilu(A);
```

provides the ILU factorization with no fill-in.

1.2 Exercises

1. Implement the GMRES method as a MATLAB function following the Algorithm in the slides.

   The syntax of the function should be the following:

   ```
   function [x,iter,resvec,flag] = mygmres(A,b,tol,maxit,x0)
   ```

   where the output parameters are

   - `x` is the solution vector
   - `iter` the number of iterations employed
   - `resvec` the vector with the norm of the residuals
   - `flag` a variable signaling breakdown (= 0: canonical termination, = −1: breakdown has occurred).

   The input parameters are the coefficient matrix, the right hand side, the tolerance, the maximum number of iterations and the initial vector.

2. Upload (from Moodle) the matrix `mat13041.rig`, which is stored in coordinate format. Solve the linear system $A\mathbf{x} = \mathbf{b}$ with $\mathbf{b}$ corresponding to an exact solution with components $x_i = \frac{1}{\sqrt{i}}$. Use your GMRES implementation with `tol` $= 10^{-10}$, `itmax` $= 550$, and `x0` the all zero vector. Plot the residual norm vs the number of iterations in a `semilogy` profile.

3. (a) Implement the preconditioned GMRES method as a MATLAB function with the following syntax

   ```
   function [x,iter,resvec,flag] = myprecgmres(A,b,tol,maxit,x0,ptype,
       L,U)
   ```

   where the additional parameters (with respect to the mygmres function) are

   - `ptype`. A character with possible values: 'L' (left), 'R' (right) or 'S' (split), selecting the type of preconditioning.
   - `L,U` the factors of the ILU factorization.

   Note. Use the following exit test, depending on the type of preconditioning:

   $$\begin{aligned} \|(LU)^{-1}\mathbf{r}_k\| &< TOL \cdot \|(LU)^{-1}\mathbf{b}\| && \text{LEFT} \\ \|\mathbf{r}_k\| &< TOL \cdot \|\mathbf{b}\| && \text{RIGHT} \\ \|L^{-1}\mathbf{r}_k\| &< TOL \cdot \|L^{-1}\mathbf{b}\| && \text{SPLIT} \end{aligned}$$

   (b) Using the same matrix `mat13041.rig` as in the previous exercise, compute the ILU preconditioner with drop tolerance 0.1.

   Then solve the linear system $A\mathbf{x} = \mathbf{b}$ with $\mathbf{b}$ corresponding to an exact solution with components $x_i = \frac{1}{\sqrt{i}}$. Use your preconditioned GMRES implementation with `tol`

$= 10^{-10}$, `itmax` $= 550$, and `x0` the all zero vector. For each of the three variants (left, right, and split preconditioner) display the final computed residual (last component of `resvec`) and the "true" final residual norm (computed as $\|\mathbf{b} - A\mathbf{x}\|$).

(c) Solve the same linear system as in 3.(b) using the MATLAB `gmres` function. To apply the three different types of preconditioning you can follow the instructions below:

Left: since by default MATLAB `gmres` apply left preconditioning you can use the usual call to the GMRES function

```
[x,flag,res,iter,resv] = gmres(A,b,restart,tol,maxit,L,U);
```

Right: call GMRES with $A(LU)^{-1}$ as the coefficient matrix using a function handle:

```
[x,flag,res,iter,resv] = gmres(@(x)A*(U\(L\x)),b,restart,tol,maxit);
x = U\(L\x);
```

Split:

```
[x,flag,res,iter,resv] = gmres(@(x)L\(A*(U\x)),L\b,restart,tol,maxit);
x = U\x;
```

Create a table comparing the results obtained with MATLAB `gmres` with those obtained with your own implementation in 3.(b). The number of iterations may differ by 2–3 at most between the two implementations.

4. Using the same matrix as in the previous exercise, solve the linear system $A\mathbf{x} = \mathbf{b}$ with $\mathbf{b}$ corresponding to an exact solution with all ones. Use the GMRES method with `tol` $= 10^{-12}$, `restart` $= 10, 20, 30, 50$ and the ILU preconditioner with `droptol` $= 10^{-2}$. For each of the four values of restart, display the number of total iterations, the relative residual at convergence, and the CPU time employed.

   Provide the convergence profiles for the four values of restart in the same figure.

5. Download matrix `ML_laplace.mtx` from the Moodle platform. Solve the linear system with this matrix as coefficient matrix and a right hand side computed so that the exact solution is the vector of all ones. Keeping the *restart* parameter fixed to 50 use different drop tolerances for the ILU preconditioner (`droptol` $= 2 \cdot 10^{-2}, 10^{-2}, 3 \cdot 10^{-3}, 10^{-3}, 10^{-4}, 10^{-5}$). For each of the six runs display on a table: the drop tolerance, the number of iterations, the CPU time needed for computing the preconditioner (tprec), the CPU time needed for GMRES to solve the system (tsol), the total CPU (tprec+tsol), the final residual norm and the **density** ρ of the preconditioner[1]

   Plot the convergence profile of the six runs on the same figure (use the legend command to clearly distinguish the different curves).

6. **OPTIONAL**: Implement one of the algorithms to compute a sparse approximate inverse preconditioner. The suggested options are SPAI, FSAI, AINV. Please consult the paper [1], for some hints and references. This work can be considered as the final project for the exam. If anyone is interested, please contact me.

[1] defined as $\rho = \dfrac{nnz(L) + nnz(U) - n}{nnz(A)}$, where n is the order of the matrix. Note that the nonzero number of a sparse matrix A is computed in MATLAB with `nnz(A)`.

2 Symmetric Indefinite systems

MATLAB provides the function `minres` that implements the MINRES method. The syntax is analogous to that of the PCG implementation (see the online help for the description of the parameters).

2.1 Constrained optimization problem

The solution of the following minimization problem (equality constrained quadratic programming problem)

$$\begin{aligned} &\min J(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} - \mathbf{f}^T\mathbf{x} \\ &\text{subject to } A\mathbf{x} = \mathbf{g} \end{aligned}$$

is usually solved by finding a "saddle point" of the associated *Lagrangian*:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} - \mathbf{f}^T\mathbf{x} + (A\mathbf{x} - \mathbf{g})^T\mathbf{y}.$$

To find a stationary point of $\mathcal{L}$ we must impose that $\nabla\mathcal{L} = 0$ which is equivalent to solving the linear system

$$\overbrace{\begin{bmatrix} Q & A^T \\ A & O \end{bmatrix}}^{H} \overbrace{\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}}^{\mathbf{u}} = \overbrace{\begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}}^{\mathbf{b}}.$$

where matrix $Q \in \mathbb{R}^{n\times n}$ is SPD for convex minimization problems and $A \in \mathbb{R}^{m\times n}$ is called the constraint matrix and is assumed to have full row rank. This type of linear systems are said to be of saddle point type since matrix H is indefinite. MINRES can be used to solve this kind of symmetric indefinite linear systems. Preconditioners for MINRES must be SPD and, generally for saddle point systems, they respect the block structure of the original system.
Note: Besides optimization, many other problems lead to saddle point linear systems. For instance, the Stokes problem in fluid dynamics or the discretization and linearization of the incompressible Navier-Stokes equations (in this case matrix H is nonsymmetric since the $(1,1)$ block is so). The interested reader can find more on preconditioners for saddle point problems on [2].

Exercise

7. Solve the system $H\mathbf{u} = \mathbf{b}$. The right hand side $\mathbf{b}$ must correspond to the exact solution $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)^T$ with $\mathbf{x}_1 = 10^{-6}\cdot$ `rand(n,1)`, $\mathbf{x}_2 =$ `rand(m,1)`.

   Block matrices Q (`Q.dat`) and A (`A.dat`) have to be uploaded (from the Moodle page) in coordinate format. Convert both in MATLAB sparse format and complete matrix Q whose only triangular part is stored. Use MINRES with a block diagonal preconditioner defined as

   $$M = \begin{bmatrix} E & 0 \\ 0 & S \end{bmatrix},$$

   with $E = \text{diag}(Q)$ and $S = AE^{-1}A^T$. Both application of matrix H and of the preconditioner M^{-1} must be provided as function handles (without forming the two matrices explicitly).

NOTE. To solve the system with matrix S (needed at each MINRES iteration) it is convenient to factorize it once and for all by using the command `chol` – exact Cholesky factorization. For this particular problem matrix S and even more L turn out to be excessively dense (order of 10^8 nonzeros!). To mitigate this effect a *symmetric permutation* of S must be employed. In MATLAB a number of algorithm which do this are implemented. Among them the `amd` – approximate minimum degree order is recommended:

```
p = amd(S);
% p is the permutation vector
S0 = S(p,p);
% permuted Schur complement
U = chol(S0);
% Upper triangular Cholesky factor of the permuted S
```

In the application of the preconditioner i.e. the solution of $M\mathbf{x} = \mathbf{y}$, a linear system involving S must be solved. The function `applyschur` below solves the linear system $S\mathbf{w} = \mathbf{v}$ taking into account permutation.

```
function w = applyschur(v,p,U);
n = size(U,1);
z = v(p);    % permute vector v
u = U\(U'\z); % solve S0 u = z
pinv(p) = 1:n;
w = u(pinv); % back-permute the result
```

After doing this you can invoke the `minres` function as:

`[ .... ]= minres(` $\underbrace{@(x)....}_{\text{handle to apply}H}$ `,b,tol,itmax,` $\underbrace{@(x)....}_{\text{handle to apply}M}$ `,[],x0)`

with $\mathbf{x}_0 = M^{-1}\mathbf{b}$ and tolerance `tol`$= 1e-12$. Finally, the function handle for the preconditioner should implement the following operations: $M\mathbf{x} = \mathbf{y} \longrightarrow E\mathbf{x}_1 = \mathbf{y}_1,\ S\mathbf{x}_2 = \mathbf{y}_2$. Good luck!

References

[1] M. BENZI, *Preconditioning techniques for large linear systems: a survey*, J. Comput. Phys., 182 (2002), pp. 418–477.

[2] M. BENZI, G. H. GOLUB, AND J. LIESEN, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.