Lecture 4: Restarted GMRES + Preconditioning

Dr Qianqian Yang

School of Mathematical Sciences

Queensland University of Technology

Outline:

- GMRES recap
- Restarted GMRES
- Preconditioning
  - Introduction
  - Left versus right preconditioning
  - How to choose an appropriate preconditioner
  - Examples

GMRES Recap

In the last lecture, we used GMRES to solve $Ax = b$.

- It minimises the residual norm $\|r^{(m)}\| = \|b - A\,x^{(m)}\|$.
- Using the Arnoldi decomposition $AV_m = V_{m+1}\bar{H}_m$,

$$\|r^{(m)}\| = \|\beta\, e_1 - \bar{H}_m\, y_m\|.$$

- So the n-dimensional minimisation reduces to the m-dimensional least squares problem:

$$\min \|\beta\, e_1 - \bar{H}_m\, y_m\| \text{ over } y_m \in \mathbb{R}^m\,.$$

- That is, to "solve" $\bar{H}_m\, y_m = \beta\, e_1 \;\rightarrow\; y_m = \bar{H}_m \setminus (\beta\, e_1)$.
- Our approximate solution $x^{(m)} = V_m\, y_m \in \; \mathcal{K}_m(A, b)$.

GMRES Recap

- The following inequality determines the convergence rate of GMRES

$$\frac{\|r^{(m)}\|}{\|b\|} \leq \mathrm{cond}(X) \inf_{\tilde{p}_m \in P_m} \sup_{z \in \sigma(A)} |\tilde{p}_m(z)|,$$

  where $\tilde{p}_m(z) = 1 - z\, q_{m-1}(z)$ is a polynomial of degree m with constant coefficient 1 ($\tilde{p}_m(0) = 1$), and $A = X\ \mathrm{diag}(\lambda_i)\ X^{-1}$.
- We see that small eigenvalues are not desirable and clustering of eigenvalues is desirable.

Restarted GMRES

- Recall that GMRES builds an orthonormal basis $[v_1, v_2, \ldots, v_m]$ for the Krylov subspace $\mathcal{K}_m$, growing by one vector at each iteration.
- Each basis vector v_i is of dimension n: the size of the full problem.
- Hence storage cost for V_m is $O(mn)$, which for large m is easily the dominant storage cost (much more memory than a typical sparse matrix with $O(n)$ nonzeros).
- Furthermore, a typical sparse-matrix vector product requires $O(n)$ operations: using each entry of A once.
- Whereas the cost of orthgonalising the next Krylov basis vector against all the previous is $O(m)$, and there are $O(m)$ iterations: $O(m^2)$ work.
- For large m this can be the dominant runtime cost.

Restarted GMRES

- So for reasons of storage limitations, or of runtime considerations, there may be an upper limit on the feasible size m.
- It is quite possible that GMRES cannot converge within limited m iterations.
- One idea is to use restarting: simply take m iterations, compute $x^{(m)}$ as usual, and use this as the initial guess for a new cycle of GMRES.
- Continuing this process, restarting every m iterations, is called Restarted GMRES, denoted GMRES(m).

Restarted GMRES

How to set up initial guess in GMRES?

- Standard GMRES finds $x^{(m)} \in \mathcal{K}_m(A, b)$, effectively taking $x^{(0)} = 0$.
- If an initial guess is available, one can instead use the affine space $x^{(m)} \in x^{(0)} + \mathcal{K}_m(A, b)$.
- Let $r^{(0)} = b - Ax^{(0)}$ and set $x = x^{(0)} + \delta x$, with δx to be determined.

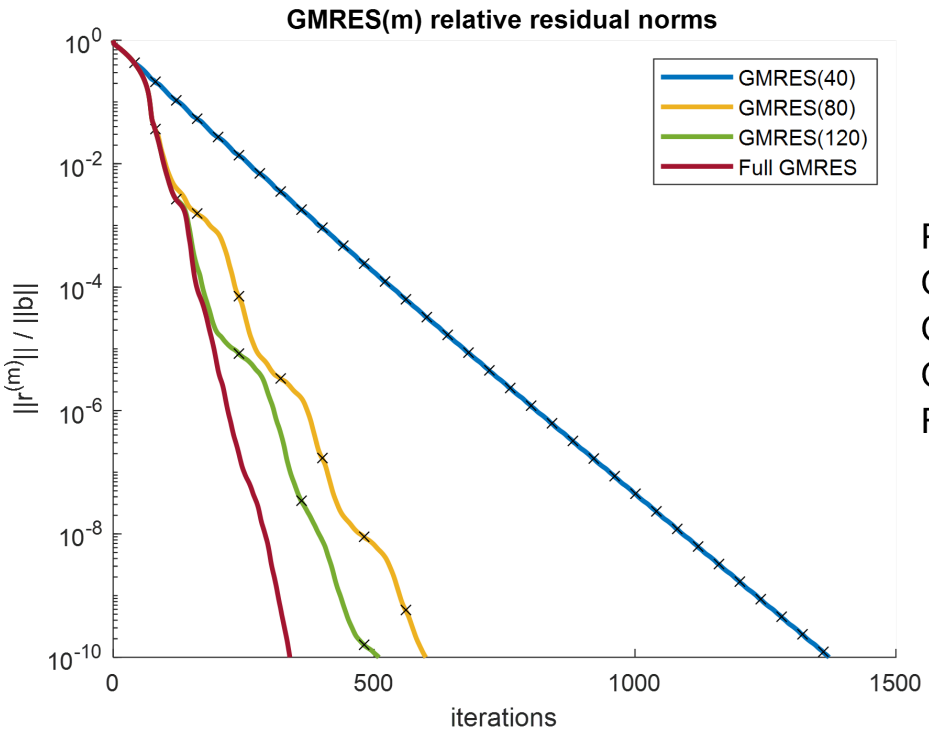
$$Ax = b \;\Rightarrow\; A(x^{(0)} + \delta x) = b$$
$$A\delta x = b - Ax^{(0)} \Rightarrow A\delta x = r^{(0)}$$

- So run GMRES on this new problem $A\delta x = r^{(0)}$, calculating $\delta x^{(m)} \in \mathcal{K}_m(A, r^{(0)})$ and hence $x^{(m)} = x^{(0)} + \delta x^{(m)}$.
- In the next cycle of GMRES, use this $x^{(m)}$ as the initial guess.

Restarted GMRES

Example:

```
A = gallery('poisson', 100); b = rand(size(A,1),1);
tol = 1e-10; maxit = 1500;
m = 40; % restart every 40 iterations; and try m = 80 and 120
[x,flag,relres,iter,resvec] = gmres(A,b,m,tol,floor(maxit/m));
```



Run time:
GMRES(40): 1.01s
GMRES(80): 0.82s
GMRES(120): 0.96s
Full GMRES: 2.02s

Restarted GMRES

In summary,

- Restarted GMRES keeps the memory requirements bounded.
- It also limits the orthgonalisation costs.
- But there are serious consequences:
  - We lose optimality across cycles: the residual is only minimised within each cycle.
  - Whatever progress had been made in localising troublesome eigenvalues is lost; the cycle starts over from scratch, albeit with a (hopefully) improved initial guess.
  - The rate of convergence may seriously deteriorate compared to full GMRES.

Preconditioning: Introduction

- In the last lecture,
  - we saw that the eigenvalue distribution plays a significant role in the convergence of GMRES,
  - in a few examples we were able to vastly improve the convergence rate by shifting the spectrum, so that instead of solving $A\,x = b$ we were solving $(A + \sigma\, I)x = b$ for some shift σ.
- This, it has to be said, is cheating. But the idea is intriguing.
- What if we could find a way to alter the spectrum, in our favour, such that the solution was unchanged?
- This is exactly the goal of preconditioning!

Preconditioning: Introduction

- The basic idea of preconditioning is to find a matrix M which approximates A in some sense ($M \approx A$), and whose inverse is computationally efficient to apply.
- Reformulate our problem

$$A\,x = b \qquad (1)$$

by multiplying by M^{-1} on both sides:

$$M^{-1}\,A\,x = M^{-1}\,b \qquad (2)$$

- Now the convergence will depend on the properties of $M^{-1}\,A$ instead of those of A.
- If we've chosen our preconditioner M wisely, the matrix $M^{-1}\,A$ will have a much more favourable spectrum (e.g. removal of small eigenvalues, clustering of eigenvalues), and (2) may be solved much more rapidly than (1).

How to choose an appropriate preconditioner

Preconditioned system:

$$M^{-1} A x = M^{-1} b \qquad (2)$$

- For the idea of preconditioning to be useful, we need to be able to compute the operation represented by M^{-1} efficiently.
- If M very closely approximates A , then $M^{-1} A \approx A^{-1} A$ is almost the identity I, but applying the preconditioner may be as hard as solving the original problem, and nothing has been gained.
- If $M = I$, then the preconditioned system (2) is the same as the original problem, so applying the preconditioner accomplishes nothing.
- So we need a matrix M that strike a balance between these two extremes.
- The matrix M has to somehow be effective at improving the spectrum of $M^{-1} A$, but not such that the expense of calculating $M^{-1} v$ is too costly.

Left versus right preconditioning

- The approach we've described so far is known as left preconditioning, since we multiply by the preconditioner on the left.
- Alternatively we can multiply on the right, to implement right preconditioning:

$$A\,M^{-1}\,M\,x = b$$

let $u = M\,x$ and solve

$$A\,M^{-1}\,u = b, \qquad (3)$$

then compute $x = M^{-1}\,u$.

Left versus right preconditioning

The difference between these two approaches is more than cosmetic.

- When preconditioning on the left, the residual vector, whose norm is monitored for convergence in GMRES, is

$$r_{\text{left}}^{(m)} = M^{-1} b - M^{-1} A x^{(m)} = M^{-1}(b - A x^{(m)}) = M^{-1} r^{(m)}$$

  where $r^{(m)}$ is the ordinary residual vector (i.e. without preconditioning).

- If M^{-1} is quite close to A^{-1} (in its effect on $r^{(m)}$), then the left-preconditioned residual approximates the error $\epsilon^{(m)}$:

$$\begin{aligned} \varepsilon^{(m)} &= x - x^{(m)} = A^{-1} b - x^{(m)} = A^{-1}(b - A x^{(m)}) \\ &= A^{-1} r^{(m)} \approx M^{-1} r^{(m)} = r_{\text{left}}^{(m)}. \end{aligned}$$

- However, if M^{-1} is not very much like A^{-1}, then it may not be clear what the left preconditioned residual really represents.

Left versus right preconditioning

- On the other hand, with right preconditioning, the residual is unaffected:

$$r_{\text{right}}^{(m)} = b - (A\,M^{-1})u^{(m)} = b - A\,M^{-1}\,Mx^{(m)} = b - A\,x^{(m)} = r^{(m)}.$$

- Hence, the residual norm $\|r_{\text{right}}^{(m)}\|$ can be monitored during the GMRES iterations with full confidence that it represents the true, unpreconditioned, residual norm.
- Note: MATLAB's inbuilt gmres function implements left preconditioning.

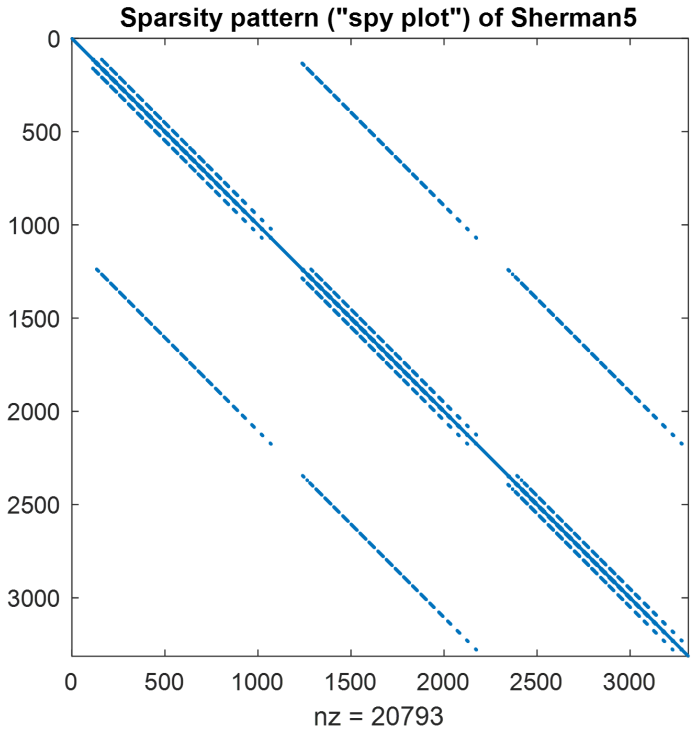
Classical Preconditioners

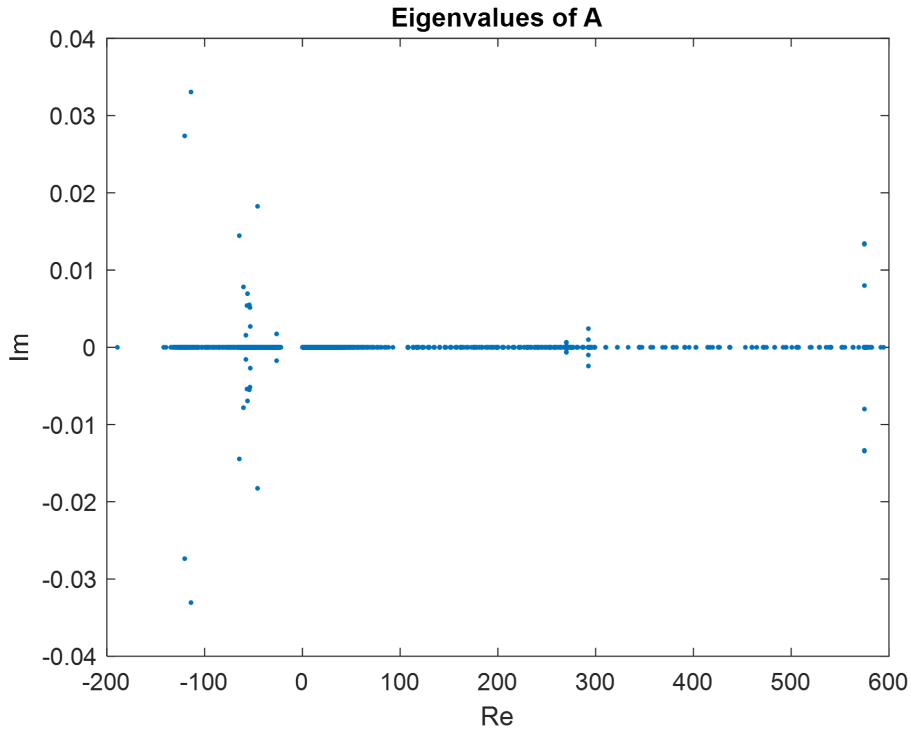
- Jacobi
- Gauss-Seidel
- Symmetric Gauss-Seidel
- Incomplete LU

To illustrate the effect of these preconditioners, we will consider a matrix from a problem in oil reservoir simulation. This is the matrix Sherman5 taken from the Matrix Market, an online repository of test matrices.

Classical Preconditioners

Sherman5: size 3312×3312; real non-symmetric matrix

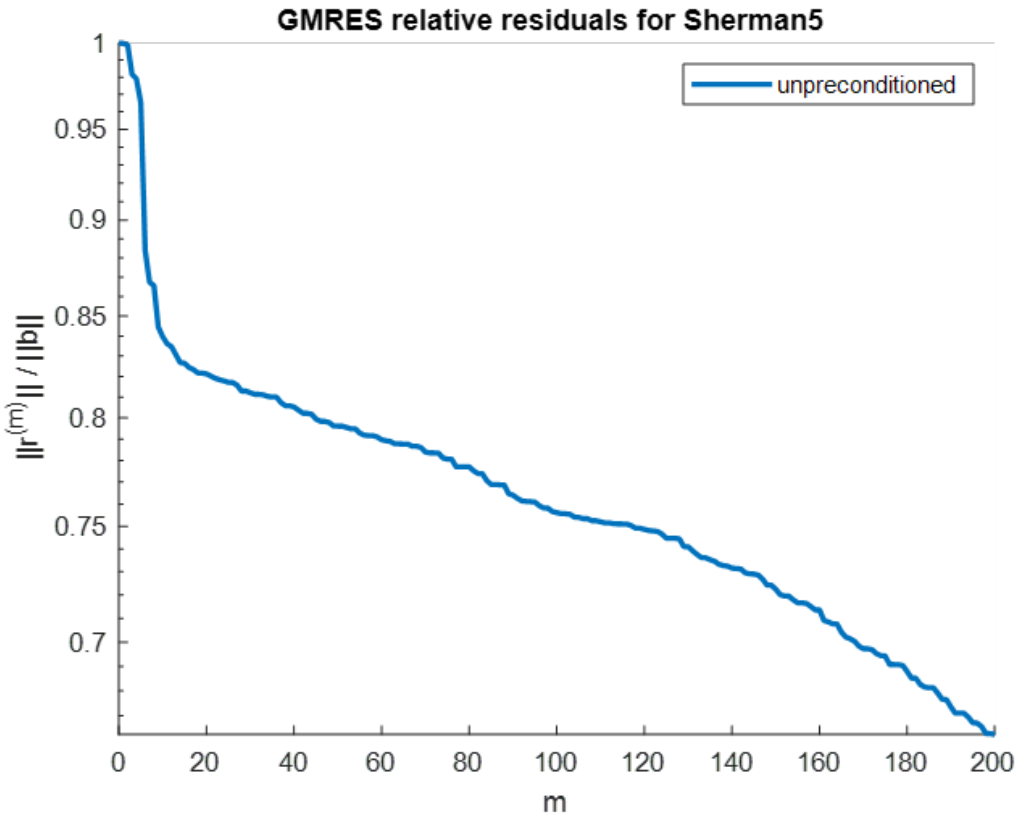




- The real parts of eigenvalues range from -200 to 600 with no clustering.
- There are some small eigenvalues in magnitude.

Classical Preconditioners

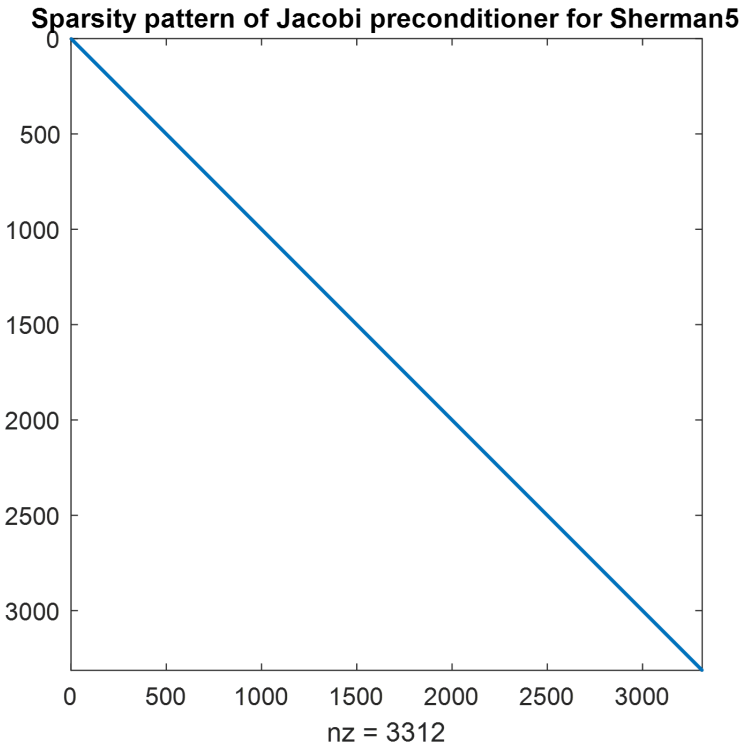
Solve the unpreconditioned system: set tolerance of 10^{-10} in relative norm with the maximum number of iterations 200. Use MATLAB's builtin function `gmres`.

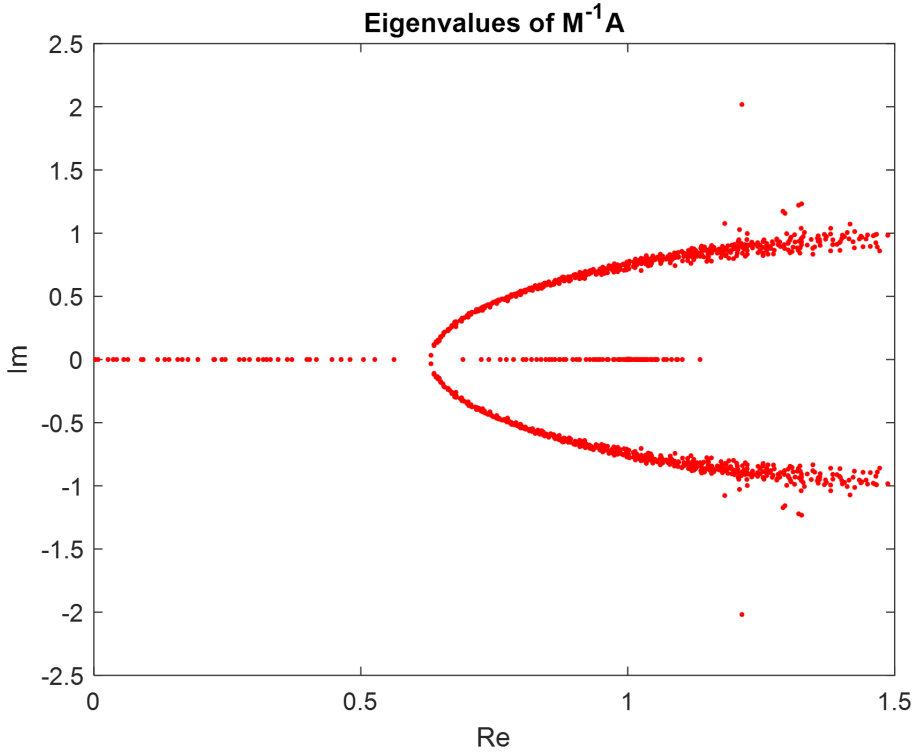


- There's really nothing much happening here; the residual norm has barely budged in 200 iterations. We need preconditioners!

Classical Preconditioners: Jacobi

Solve the preconditioned system $M^{-1} A x = M^{-1} b$ with the Jacobi preconditioner: $M = \mathrm{diag}(A)$. Cheap to apply.

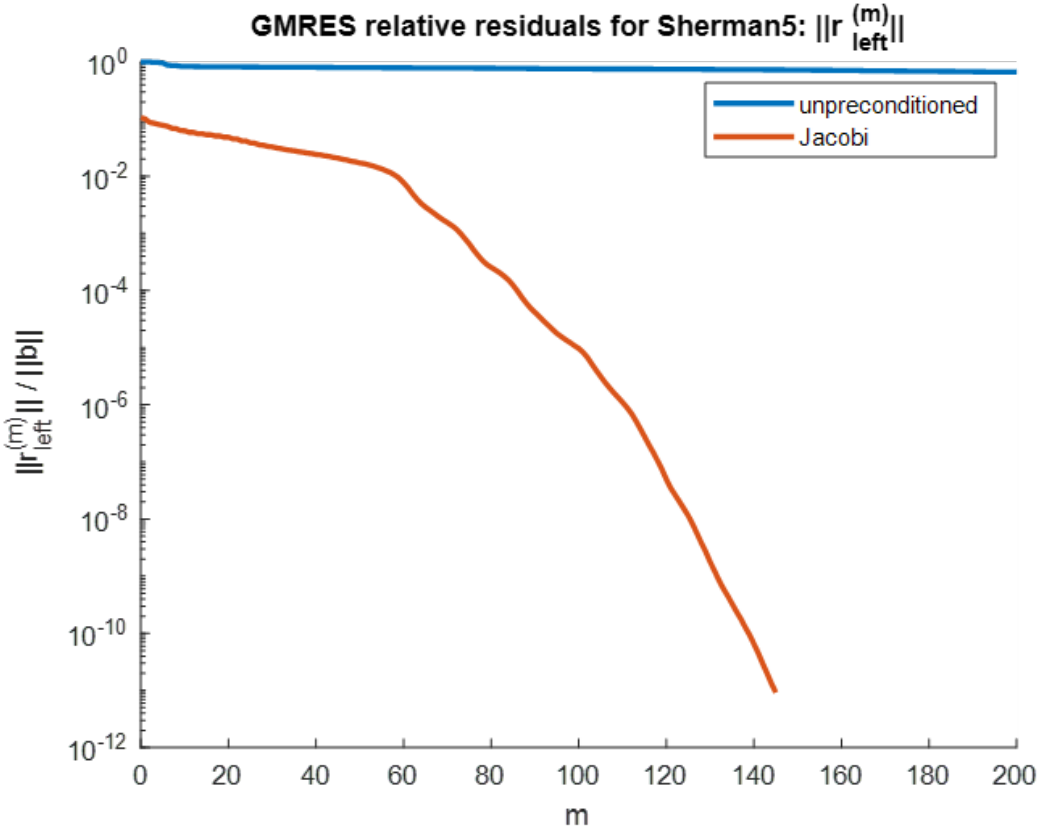




- There are no longer any eigenvalues in the left half-plane.
- The range of magnitudes has also been greatly reduced, and there is a hint of clustering around unity.
- Some small magnitude eigenvalues evidently remain.

Classical Preconditioners: Jacobi

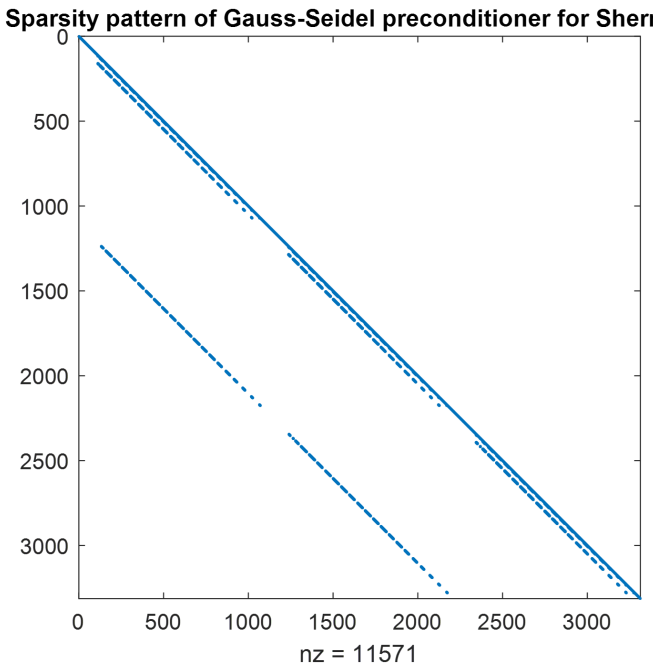
Let's check out the performance of the Jacobi preconditioner.

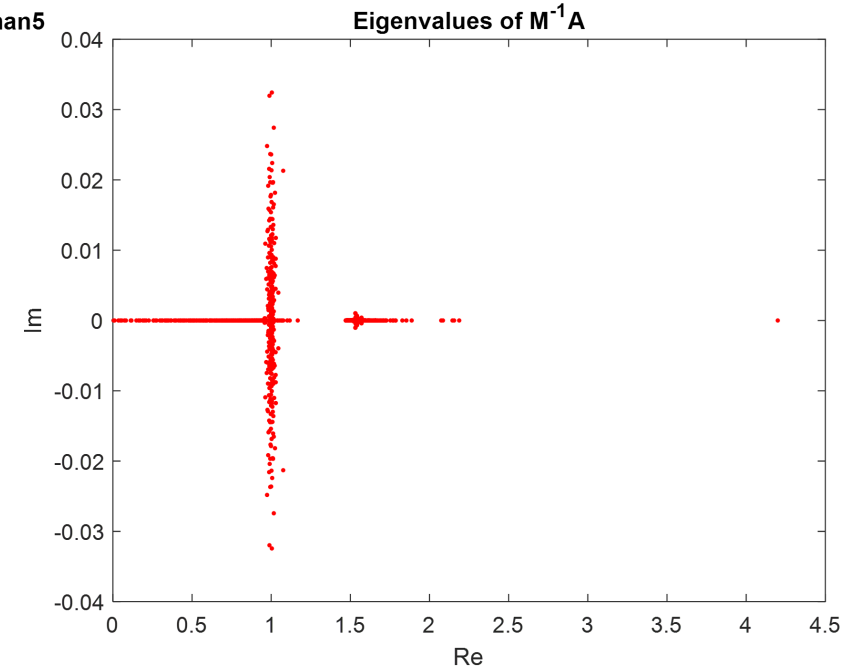


- The Jacobi preconditioned GMRES converges to the desired tolerance in 145 iterations.
- If it can be this effective, how much better could we get by choosing something a little more elaborate?

Classical Preconditioners: Gauss-Seidel

Solve the preconditioned system $M^{-1}\,A\,x = M^{-1}\,b$ with the Gauss-Seidel preconditioner: $M = \mathrm{tril}(A)$, which uses the entire lower triangular portion of A as M.

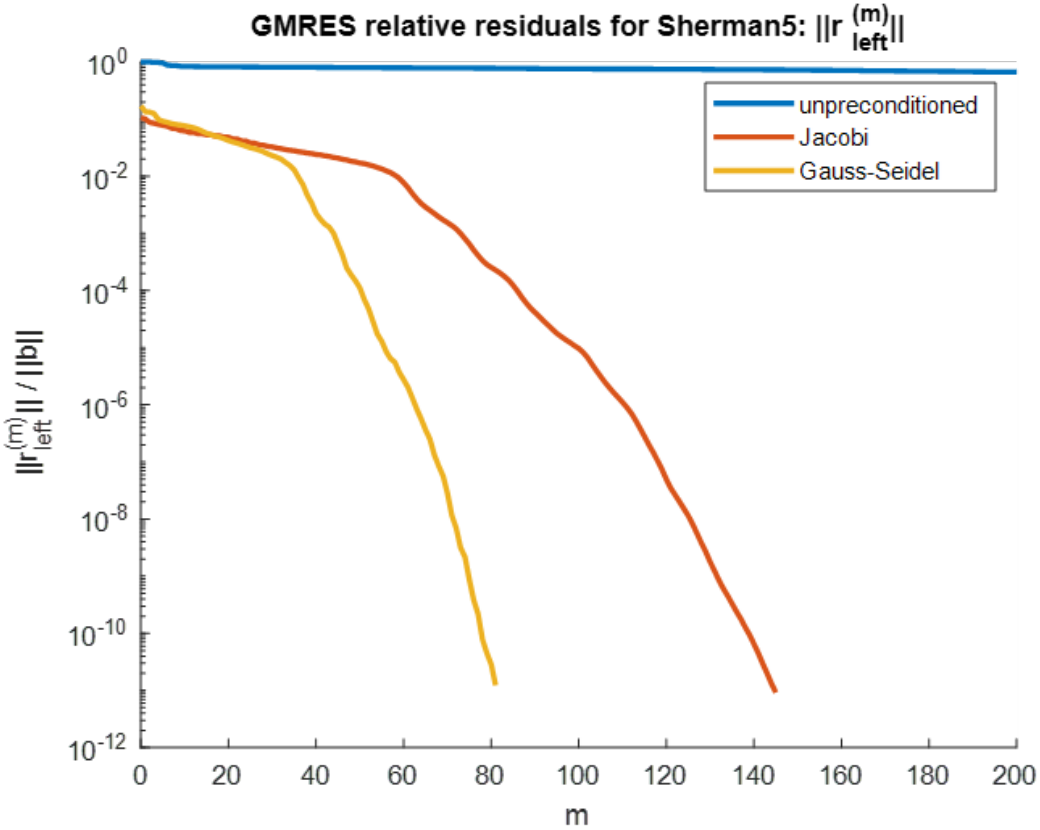




- We can see a definite clustering of eigenvalues around unity, and again the negative eigenvalues are all gone. There are a couple of small eigenvalues still hanging around.
- This looks like a better distribution than we had for Jacobi, so we might expect convergence to be improved again.

Classical Preconditioners: Gauss-Seidel

Let's check out the performance of the Gauss-Seidel preconditioner.



- By including more information about A in our preconditioner M, the convergence has been improved again.

Classical Preconditioners: Symmetric Gauss-Seidel

Solve the preconditioned system $M^{-1}\,A\,x = M^{-1}\,b$ with the symmetric Gauss-Seidel preconditioner:

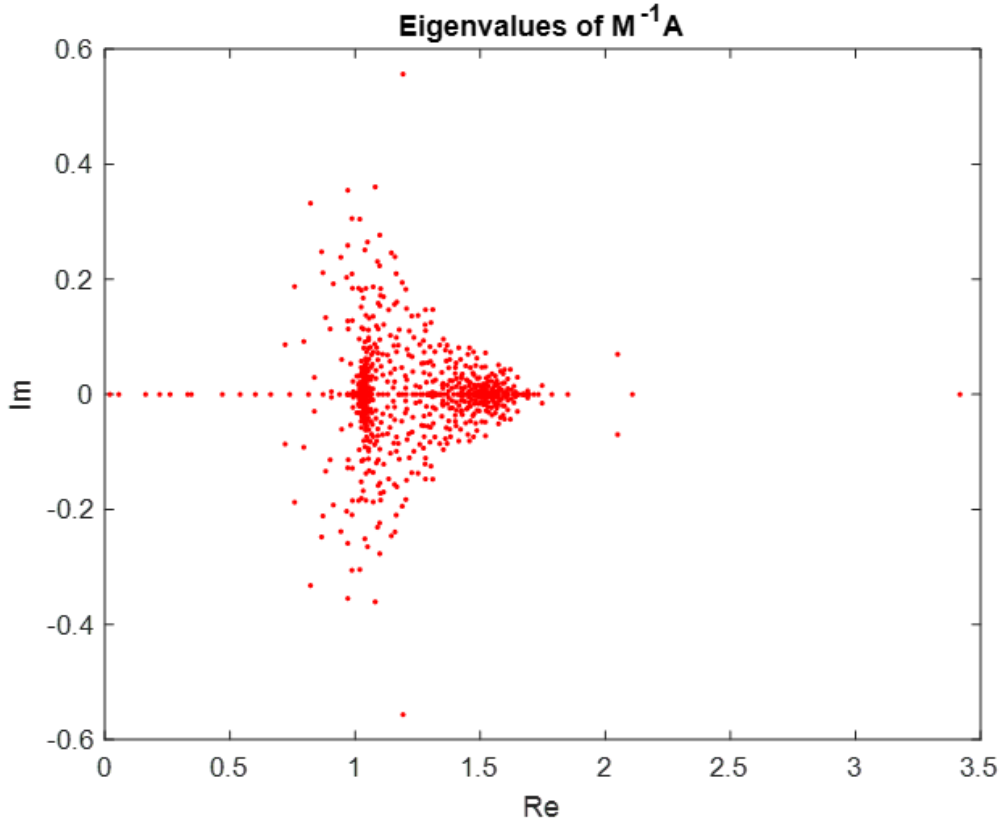
$$M = M_1\,M_2$$

with $M_1 = \mathrm{tril}(A)$ and $M_2 = \mathrm{triu}(A)./(\mathrm{diag}(A))$.

- It brings in the upper triangular portion of A as well, through the product $M = M_1\,M_2$, where M_1 is the lower triangular portion and M_2 is the upper triangular portion scaled by the diagonal (to avoid "double-counting" the diagonal).
- The product $M = M_1\,M_2$ has a similar sparsity pattern to A.
- Importantly, this product is never computed explicitly. Whenever the algorithm calls for the product $w = (M_1\,M_2)^{-1}\,A\,v$ for some vector v, it is computed entirely through matrix-vector operations: $w = M_2^{-1}\,(M_1^{-1}\,(A\,v))$, i.e., $M_2 \backslash (M_1 \backslash (A\,v))$.

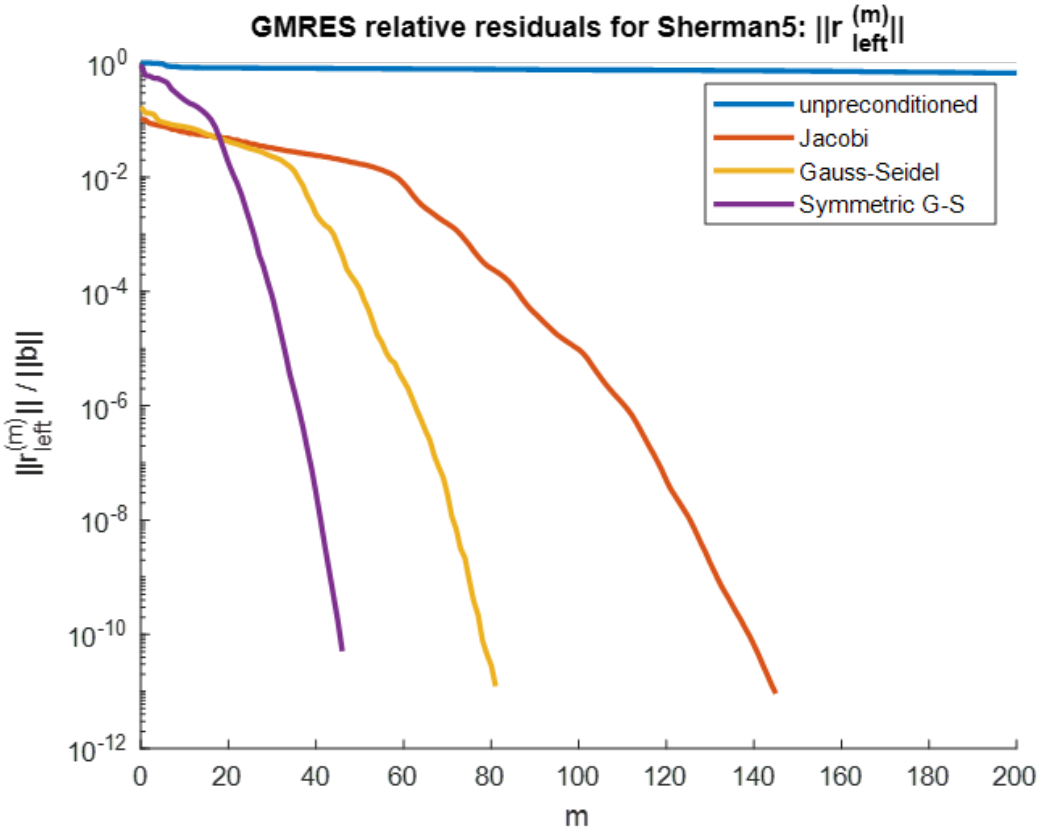
Classical Preconditioners: Symmetric Gauss-Seidel

Let's have a look at the eigenvalue spectrum for the symmetric Gauss-Seidel preconditioned matrix. Note that here we did compute the product $(M_1\, M_2)^{-1}\, A$, but this is just for the purposes of illustration.



- This may or may not look better than ordinary Gauss-Seidel to you, but certainly it's quite similar in its effect on the spectrum.

Classical Preconditioners: Symmetric Gauss-Seidel



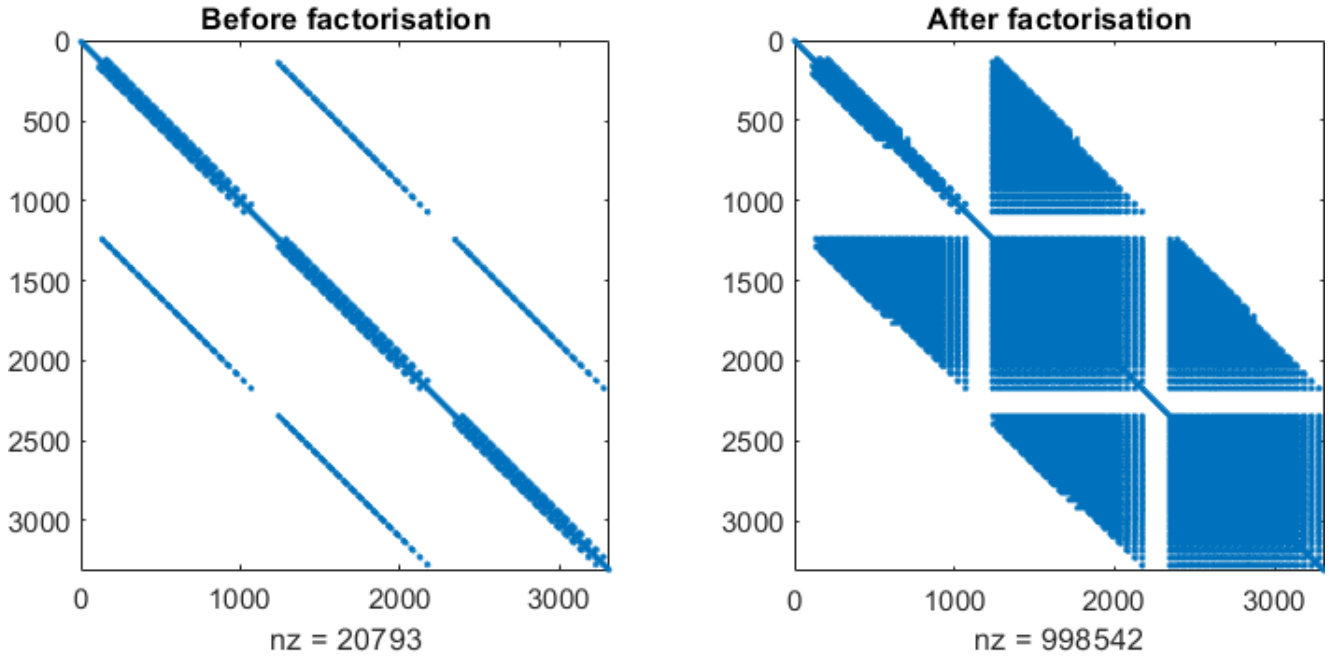
- Symmetric Gauss-Seidel does outperform ordinary Gauss-Seidel as a preconditioner for this problem.

Classical Preconditioners: Incomplete LU factorisation

- This is an example of a preconditioner that is often used in practice.
- The idea of incomplete LU is to compute an approximate factorisation for A, $A \approx L\,U$, where the LU factors are not the exact LU factors (which would be too expensive to work with) but inexact, sparse LU factors.

Classical Preconditioners: Incomplete LU factorisation

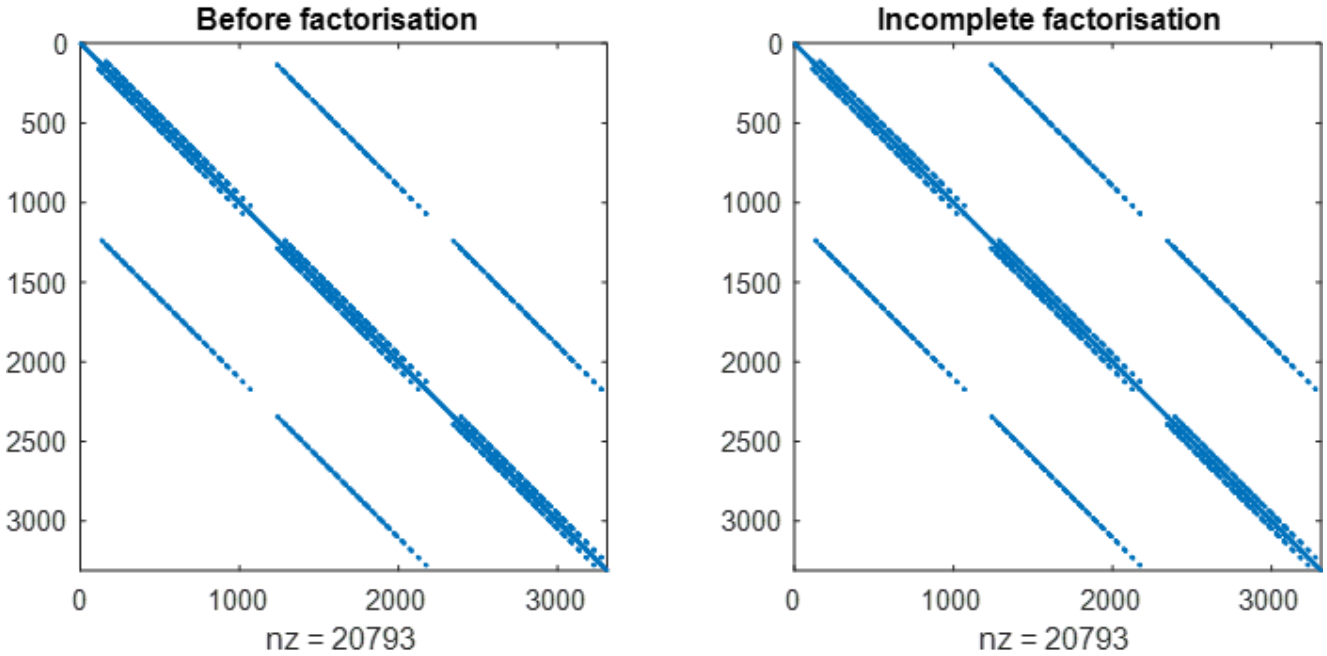
For comparison, let's first consider the full, exact LU factorisation of our matrix.



- Notice the large amount of fill-in generated by this factorisation.
- The number of nonzero entries in the factorised matrix has exploded: almost 50 times more nonzero entries after factorisation.
- The computational and memory costs associated with generating this factorisation quickly get out of control for larger matrices, making it impractical.

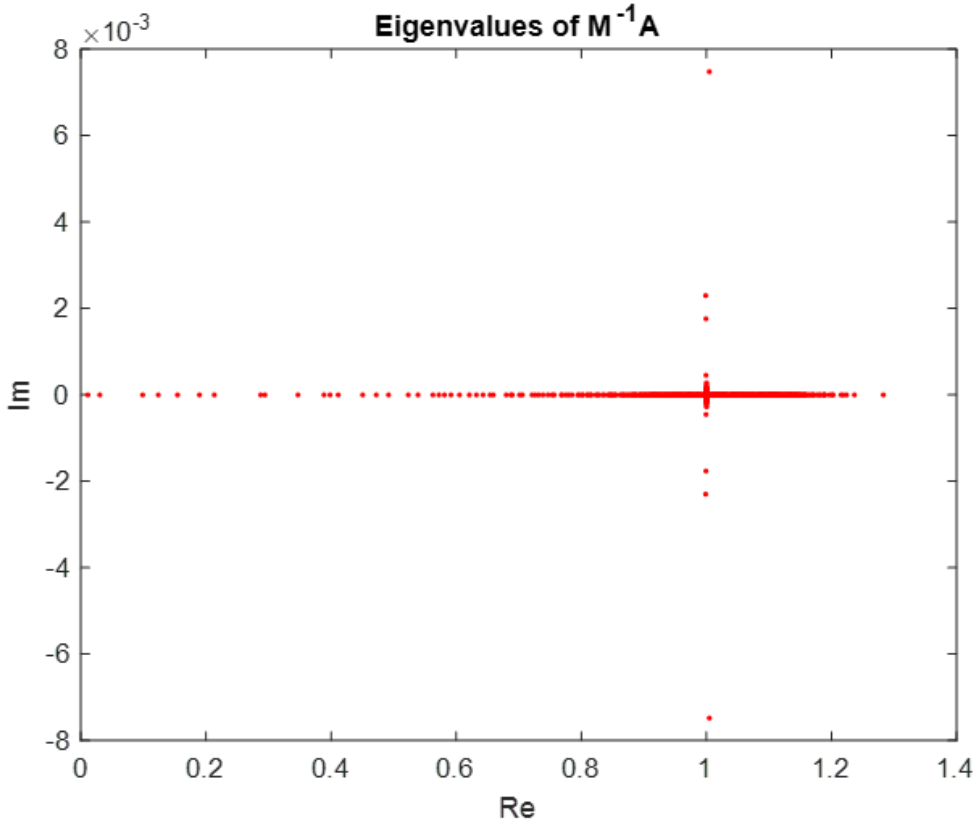
Classical Preconditioners: Incomplete LU factorisation

- Idea of incomplete LU (ILU): Drop entries from the factorisation to keep the matrix sparse.
- The computed LU factors are no longer exact, but in many cases they suffice as a preconditioner.
- The criteria for dropping entries can be simple, or more elaborate based on tolerances.
- Here we'll just look at ILU(0), which drops every single entry that would have otherwise filled in.



Classical Preconditioners: Incomplete LU factorisation

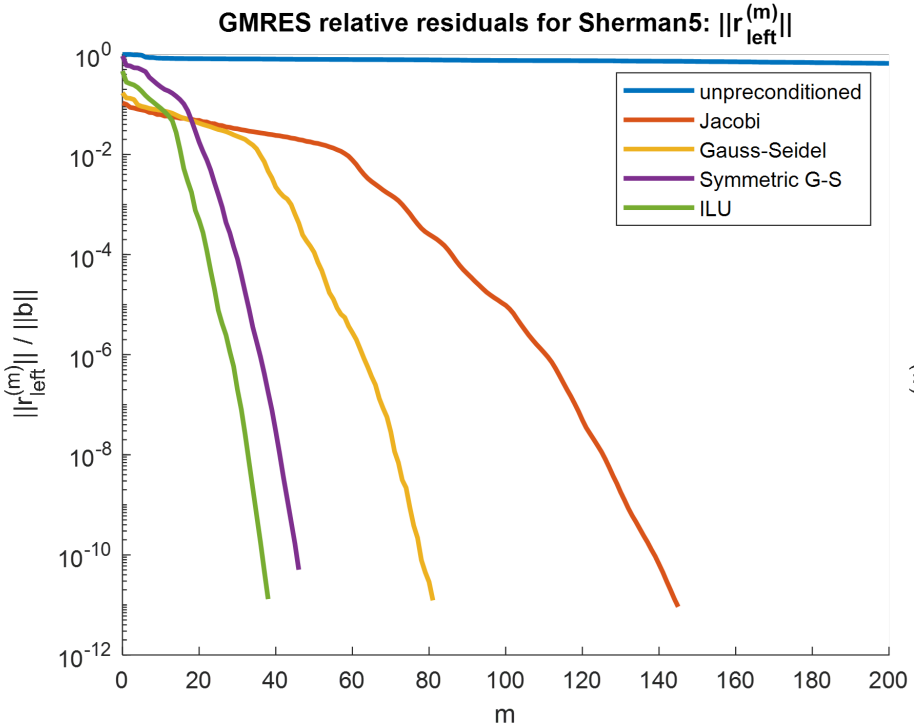
Let's check out the resulting spectrum.

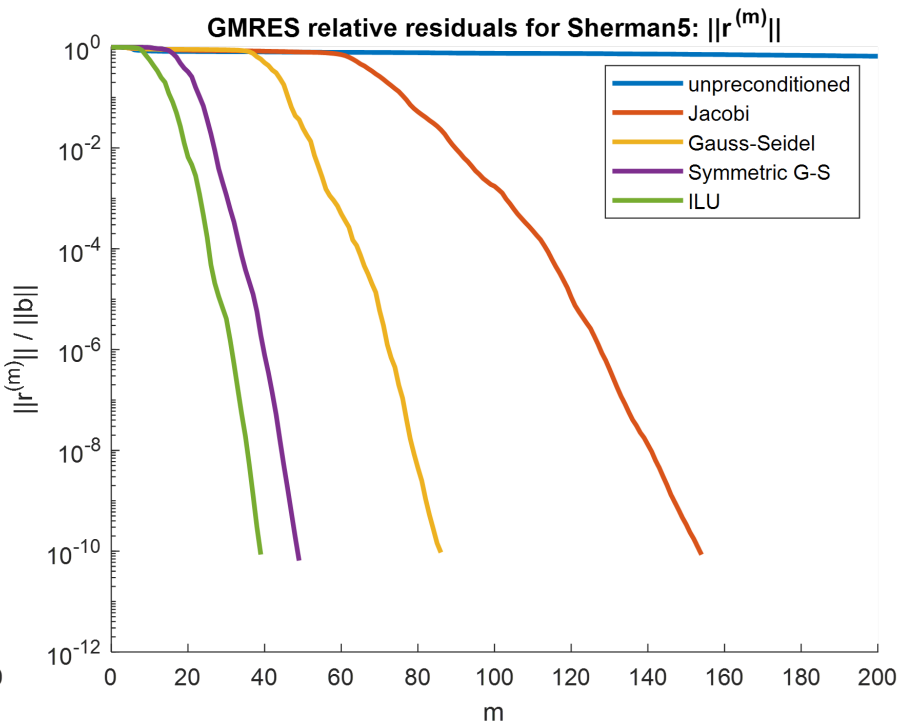


- The clustering around unity looks very nice.
- There are still those couple of pesky small eigenvalues that none of our preconditioners has been able to properly deal with.

Classical Preconditioners: Incomplete LU factorisation

Let's check out the performance of ILU.





- ILU is the winner for today. It outperforms all our other preconditioners, yielding convergence in only 38 iterations.
- Note: the plot on the left is using $||r_{\text{left}}^{(m)}|| = ||M^{-1}r^{(m)}||$ which is from the preconditioned system; the plot on the right is using the true residual $||r^{(m)}||$.

Summary of Lectures 3 and 4

GMRES (Generalised Minimal Residual)

- Iterative Krylov subspace method for solving $Ax = b$.
- Finds the approximate solution $x^{(m)} \in \mathcal{K}_m(A, b)$ by minimising the residual norm $||r^{(m)}||$.
- Convergence rate depends on the properties of the matrix A: the $\mathrm{cond}(X)$ and the eigenvalue distribution of the matrix A.

Restarted GMRES (GMRES(m))

- Restarts after every m steps to control storage cost and run time.
- May have slow convergence or even stall

Preconditioning

- To improve the convergence for iterative methods.
- Left-preconditioned system: $M^{-1}\,A\,x = M^{-1}\,b$.
- Right-preconditioned system: $A\,M^{-1}\,u = b$, then $x = M^{-1}\,u$.
- Never form M^{-1} or $M^{-1}\,A$ explicitly; always compute the action of M^{-1} through $M^{-1}v$ (i.e., $M\backslash v$).