Project Proposal
CS 575: Numerical Linear Algebra
Spring 2023

John Tran

April 4, 2023

Topic

The main topic is to explore the code implementation of the Generalized Minimal RESidual method (GMRES) applied with a simple preconditioner (usually to reduce the condition number of the problem and speed up convergence – i.e., with fewer iterations).

The linear system investigated for this project will be the steady-state solution (time independent) advection-diffusion problem of the form

$$au'(x) = \kappa u''(x) + \psi(x)$$
$$u(0) = \alpha, u(1) = \beta$$

as a 2-point boundary value problem with Dirichlet boundary conditions α, β on the interval $[0, 1]$ for x (equation (2.82) in [1])

We can rewrite the above equation by dividing both sides by a and gathering the derivatives of u on the left-hand side

$$\epsilon u''(x) - u'(x) = f(x)$$

where we let the inverse of the Péclet number $\epsilon = \kappa/a$ and $f(x) = \psi(x)/a$ (equation (2.88) in [1])

However, since the main focus will be on implementing GMRES and exploring the relative performance with other methods (discussed more in the Code and Implementation Details section), we can use a simplified version of the advection-diffusion equation above mentioned in [2] (equation (2.20) where $L = 1$). The advection-diffusion equation is also referred to as the convection-diffusion equation in [2]

$$-au''(x) + bu'(x) = 0, \quad 0 < x < 1$$
$$u(0) = 0, u(1) = 0 \quad \text{(typo in book where } u(1) = 1\text{)}$$

where there is some constant for both u'' and u' this time along with some substitutions for $f(x) = 0$ and the boundary constants $\alpha = 0, \beta = 1$ from above.

Given this simplification, we are given the exact solution [2]

$$u(x) = \frac{1 - e^{Rx}}{1 - e^{R}}$$

where the Péclet number $R = b/a$

If we look at the linear differencing equation for solving the linear equation $Ax = y$ for $A \in \mathbb{R}^{m \times m}$ (b replaced with y to prevent confusion with the b constant above in front of u') where $y = f(x) = 0$ for our simplification, we can use the centered schemas for both the first and second order derivatives of u

$$u' = \frac{u_{i+1} - u_{i-1}}{2h}$$
$$u'' = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}$$

where h is the mesh step size ($h = 1/(m+1)$) and plugging these derivatives above,

$$-au''(x) + bu'(x) = 0$$
$$-a\Big(\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}\Big) + b\Big(\frac{u_{i+1} - u_{i-1}}{2h}\Big) = 0$$

Note: the forward and backward schema were discussed for the first order derivative during office hours, but the centered scheme will be used for now and the benefits of the forward and backward schema will be looked at later (may not be needed for specific choices for b)

So, in order to construct the matrix A, we can group terms by u_{i-1}, u_i, u_{i+1} for values $i \in \{1, 2, \ldots, m\}$, remembering we have the boundary values $u_0 = u_{m+1} = 0$ so no extra vector is needed to account for them.

$$-(1-c)u_{i+1} + 2u_i - (1+c)u_{i-1} = 0$$

as shown in [2] (equation (2.21)) where $c = Rh/2$. So, we have the form of A with this simplification as a tridiagonal matrix with $-(1-c)$ along the superdiagonal, 2 along the main diagonal, and $-(1+c)$ along the subdiagonal.

These are just some of the main details from [2] that will be adapted from the implementation (the exact solution is also given, discussed in the Code and Implementation Details section). One important thing to note is since b in $Ax = b$ is chosen to be 0, A would have to be nonsingular to give a unique solution. However, we could arbitrarily choose some other function for $f(x) = b$ to avoid this (and make the solution a little more interesting).

There are issues discussed in [2] where diagonal dominance of the tridiagonal A is lost for specific values of c (and thus b) for the central schema case for u'. However, this should not be an issue when c is chosen to be less than 1 (e.g., simple when a and b are chosen to be similar in magnitude where as a reminder, $c = Rh/2 = bh/2a$, and h is chosen to be sufficiently small).

From here, the linear system is established and it is a matter of coding up GMRES to solve this system (and experimenting with different ways to implement GMRES and choice of parameters, discussed in the next section).

Potential Sources

All the potential sources that I have found of interest will be found at the end of the proposal in the References section. Due to time constraints, only brief details are discussed for GMRES and potential preconditioners.

For information on setting up the initial linear system via finite differencing mentioned above, [1] and [2] were quite heavily referenced, as mentioned in the previous section.

For information on GMRES and possible implementation details, [2] [3] [4] cover a wide range of details (where [4] covers a specific implementation in `Julia`). The main approach of GMRES is approximating the solution of $Ax = b$ of x with $x_n \in K_n$, the n-th Krylov subspace spanned by $\{r_0, Ar_0, A^2r_0, \ldots, A^{n-1}r_0\}$ where r_0 is the initial error to some initial guess x_0, minimizing the residual $r_n = b - Ax_n$. To compute the basis of K_n, the Arnoldi iteration is used instead of the spanned vectors to find an orthonormal basis. Then through the Arnoldi process, we can find a Hessenberg matrix to find an intermediate vector y_n to minimize the final residual $r_n = \tilde{H}_n y_n - \beta e_1$ where β is the 2-norm of the initial error r_0. Finally, we can compute the final solution vector x_n with the orthogonalization method for the Hessenberg matrix used to find the orthonormal basis V_m by adding the initial guess x_0 with the product of V_m and the minimized vector y_0. Some details were left out but more time will be needed to go through the finer aspects of the Arnoldi iteration and GMRES.

Note: it is interesting to see that the author of [2] was one of the people that helped develop the GMRES algorithm, so that will certainly be a useful reference form GMRES in general.

For information on preconditioning, [5] [6] [7] [4] [2] would all provide great background on preconditioners, especially [4] [2] for the case of GMRES. There are different types of preconditioned systems (e.g., left, right, two-sided) and [5] [2] (Chapter 9.3) talks about choice comparison of the preconditioner types. As for the preconditioner choice itself, various sources outline the main three preconditioners: Jacobi, Gauss-Seidel, and Successive Over Relaxation (SOR) with SOR having an extra ω paramter to tune.

Code and Implementation Details

For the code portion, I was thinking of comparing the performance between the following methods:

- LU factorization
- Conjugate Gradient (CG)
- GMRES
- Preconditioned GMRES
- Preconditioned CG (if time permits)

As discussed in office hours, the LU factorization could be used as a baseline to calculate the error with each iteration of the Krylov subspace methods (and thus determine the convergence). However, since the exact solution to either sets of boundary conditions mentioned above (fixed or variable α and β for Dirichlet boundary conditions) is known from [2] and [1], the exact solution may be used instead of the approximation given by LU factorization. Either way, since the main focus of the project is on GMRES, built-in functions will be used to find the LU factorization.

Then for the other two main methods (CG and GMRES), their algorithms will be coded up from scratch (CG should be fairly straightforward).

Since I am more familiar with `Python`, that is what I will be using to code up the methods and plot the convergence with errors and runtime (with and without preconditioning). The linear system mentioned above will be sparse (some tridiagonal matrix for A) so sparse formats will be looked into (e.g., using $scipy.sparse$ with maybe CSR format).

In summary, a lot of the implementation details (like choice of parameters and preconditioner) are still undecided but feedback on which combination might work best to start off with would be appreciated.

References

[1] R. J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations*. Society for Industrial and Applied Mathematics, 2007. DOI: 10.1137/1.9780898717839. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9780898717839.

[2] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Second. Society for Industrial and Applied Mathematics, 2003. DOI: 10.1137/1.9780898718003. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9780898718003.

[3] Wikipedia contributors. "Generalized minimal residual method — Wikipedia, the free encyclopedia." (2022), [Online]. Available: https://en.wikipedia.org/w/index.php?title=Generalized_minimal_residual_method&oldid=1126405909.

[4] J. Verschelde. "Krylov and Preconditioned GMRES." (Oct. 2022), [Online]. Available: http://homepages.math.uic.edu/~jan/mcs471/krylov.pdf.

[5] Wikipedia contributors. "Preconditioner — Wikipedia, the free encyclopedia." (2023), [Online]. Available: https://en.wikipedia.org/w/index.php?title=Preconditioner&oldid=1141155872.

[6] G. Fasshauer. "Preconditioning." (2006), [Online]. Available: http://www.math.iit.edu/~fass/477577_Chapter_16.pdf.

[7] A. J. Wathen, "Preconditioning," *Acta Numerica*, vol. 24, pp. 329–376, 2015, Found online copy here at the provided url. DOI: 10.1017/S0962492915000021. [Online]. Available: http://people.maths.ox.ac.uk/wathen/preconditioning.pdf.