Iterative solvers for linear systems
APPM 4600 Project
University of Colorado Boulder
Department of Applied Mathematics

1 Project Summary

The cost of solving a linear system of the form $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is the step that often dominates the cost of in most algorithms. This is because if $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, the cost of inverting $\boldsymbol{A}$ directly cost $O(n^3)$. It is for this reason that in practice, it is very common to use an iterative solver to construct an approximate solution of the linear system. Roughly speaking, iterative solvers create a sequence of approximations to the solution. The goal is for the sequence to converge to the solution. In class, you learned about Conjugate Gradient which is an iterative solver but only works for matrices with special structure. In this project, you will explore several more widely applicable iterative solvers. You will learn about there potential and limitations.

2 Project Background

This project is concerned with approximating the solution to

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$ via an iterative solver. These iterative solvers only require the ability to apply the matrix $\boldsymbol{A}$ to a vector. Typically, these solvers are coupled with fast matrix multiplication methods that either exploit sparsity or structure in the matrix to reduce the cost of applying the matrix to a vector from $O(n^2)$ with dense linear algebra to $O(n)$.

2.1 Richardson iteration

The first iterative solver you will investigate is called *Richard iteration*. Roughly speaking the iteration technique recast the problem as a root finding task.

For this section, assume that $\boldsymbol{A}$ is positive definite or positive semidefinite. Let $\boldsymbol{x}^*$ denote the exact solution to the linear system. This means that $\boldsymbol{A}\boldsymbol{x}^* = \boldsymbol{b}$. The Richard iteration is given by the following:

$$\begin{aligned} \text{given} \quad & \boldsymbol{x}_0 \\ \boldsymbol{x}_{k+1} \quad &= (\boldsymbol{I} - \alpha\boldsymbol{A})\,\boldsymbol{x}_k + \alpha\boldsymbol{b} \end{aligned}$$

where α is some user defined constant.

2.1.1 Questions to Investigate

1. Show that $(\boldsymbol{I} - \alpha \boldsymbol{A})\, \boldsymbol{x}^* + \alpha \boldsymbol{b} = \boldsymbol{x}^*$.

2. Define the absolute error at step $k+1$ to be $\|\boldsymbol{x}^* - \boldsymbol{x}_{k+1}\|$. What conditions are required in order for the method to converge? With these conditions how should you chose α? Are there any conditions on $\boldsymbol{x}_0$?
   Hint: Start by considering $\boldsymbol{A}$ is also symmetric and remember our fixed point iteration theory from class.

3. Test the performance of the method for a collection of matrices that satisfy the specific conditions listed for the method. For which matrices is the method faster than inverting directly? Does this depend on size, sparsity pattern, etc.?
   Hint: you will want to learn how to use the Python fast matrix multiplication toolbox.

2.2 GMRES: Generalized Minimum Residuals

The second iterative method you will consider is GMRES. This method is probably the most widely used iterative solver as it can be utilized for general matrices. While it is relatively easy to gain intiution on how to make the method converge quickly, conducting the actual analysis explaining how the method converges, etc. is still an open problem. This section will introduce you to the method and give you some exercises to gain intiution on the behavior of the method.

2.2.1 Krylov subspaces

The *Krylov subspaces* $\mathcal{K}_m$ generated by $\boldsymbol{A}$ and $\boldsymbol{b}$ are defined as follows

$$\mathcal{K}_m = \{\boldsymbol{b}, \boldsymbol{A}\boldsymbol{b}, \ldots, \boldsymbol{A}^{m-1}\boldsymbol{b}\}.$$

In other words, it the space spanned by the vectors involving powers of $\boldsymbol{A}$ being applied to $\boldsymbol{b}$.

A matrix $\boldsymbol{A}$ can be written in Hessenberg form via the following orthogonal similarity transformation

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{H}\boldsymbol{Q}^*$$

where $\boldsymbol{Q}$ is orthogonal and $\boldsymbol{H}$ is a *Hessenberg* matrix. Let $\boldsymbol{Q}_m$ denote the $n \times m$ matrix whose columns are the first m columns of $\boldsymbol{Q}$. Let $\boldsymbol{q}_j$ denote the j^{th} column of $\boldsymbol{Q}_m$. These vectors form a basis for $\mathcal{K}_m$.

Let $\boldsymbol{K}_m$ denote the matrix

$$\boldsymbol{K}_m = \left[\boldsymbol{b} | \boldsymbol{A}\boldsymbol{b} | \boldsymbol{A}^2\boldsymbol{b} | \cdots | \boldsymbol{A}^{m-1}\boldsymbol{b}\right]$$

Then the QR factorization of $\boldsymbol{K}_m$ is given by

$$\boldsymbol{K}_m = \boldsymbol{Q}_m \boldsymbol{R}_m$$

where $\boldsymbol{Q}_m$ is the matrix defined above.

2.2.2 Now to GMRES

The goal of GMRES is pretty simple. Essentially the goal is to construct $\boldsymbol{x}_k$ so that the norm of the residual vector $\boldsymbol{r}_k$ is minimized where

$$\boldsymbol{r}_k = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k.$$

This means that the linear system solve has been changed to a least squares problem. One option for finding the minimizing vector would be to find $\boldsymbol{c} \in \mathbb{R}^n$ such that

$$\|\boldsymbol{A}\boldsymbol{K}_k\boldsymbol{c} - \boldsymbol{b}\|$$

is minimized. Unfortunately, this is both unstable and expensive.

Instead the vector $\boldsymbol{x}_k$ is expressed as $\boldsymbol{x}_k = \boldsymbol{Q}_k\boldsymbol{y}$ where $\boldsymbol{y}$ is the minimizer of

$$\|\boldsymbol{A}\boldsymbol{Q}_k\boldsymbol{y} - \boldsymbol{b}\|.$$

The algorithm is given as follows (take from [**?**]) :

1. *Start* with an initial guess $\boldsymbol{x}_0$ and compute the residual $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$ and the normalized residual vector $\boldsymbol{v}_1 = \frac{\boldsymbol{r}_0}{\|\boldsymbol{r}_0\|}$.

2. *Iterate:* for $j = 1, 2, \ldots, k, \ldots$ until "satisfied" do:
   $h_{i,j} =< \boldsymbol{A}\boldsymbol{v}_j, \boldsymbol{v}_j >$ for $i = 1, 2, \ldots, j$ (Make Hessenberg matrix)
   $\hat{\boldsymbol{v}}_{j+1} = \boldsymbol{A}\boldsymbol{v}_j - \sum_{i=1}^{j} \boldsymbol{h}_{i,j}\boldsymbol{v}_i$ (project away from space already captured)
   $h_{j+1,j} = \|\hat{\boldsymbol{v}}_{j+1}\|$
   $v_{j+1} = \frac{\hat{\boldsymbol{v}}_{j+1}}{h_{j+1,j}}$

3. *Form the approximation*

   $$\boldsymbol{x}_k = \boldsymbol{x}_0 + \boldsymbol{V}_k\boldsymbol{y}_k$$

   where $\boldsymbol{V}_k$ is the matrix with j^{th} column vector $\boldsymbol{v}_j$ and $\boldsymbol{y}_k$ is the minimizer of $\|\beta\boldsymbol{e}_1 - \boldsymbol{H}_k\boldsymbol{y}\|$. Here $\beta = \|\boldsymbol{r}_0\|$, $\boldsymbol{e}_1$ denotes the first connocial vector and $\boldsymbol{H}_k$ denotes the Hessenberg matrix with entries $h_{i,j}$ defined by the algorithm.

2.2.3 Questions to investigate

1. Write a code that finds the minimizer of

   $$\|\beta\boldsymbol{e}_1 - \boldsymbol{H}_k\boldsymbol{y}\|.$$

   Here $\beta = \|\boldsymbol{r}_0\|$, $\boldsymbol{e}_1$ denotes the first connocial vector and $\boldsymbol{H}_k$ denotes the Hessenberg matrix with entries $h_{i,j}$ defined by the algorithm.

2. Come up with some ways to set a stopping criterion for GMRES.
   Convergence error, value of $h_{j,j}$ Provide motiviation for each of your potential stopping conditions.

3. Write your own GMRES code. Which of your potential stopping conditions works best and why?

4. Compare the number of iterations, accuracy, etc. of your GMRES with the GMRES from a software package. What are differences and similarities? Which would you use in practice?

5. It is pretty easy to construct a matrix where you have control over the condition number and the spectrum. Let $\boldsymbol{R}$ denote an $n \times n$ matrix with random entries. This matrix is full rank. Let $\boldsymbol{Q}$ denote the orthogonal matrix that results from the QR factorization of $\boldsymbol{R}$. (You could just use $\boldsymbol{R}$ but I like having control over my eigenvectors.) Let $\boldsymbol{D}$ denote a diagonal matrix where you get to choose the diagonal entries $\{d_1, \ldots, d_n\}$ (ordered largest to smallest. Then $\boldsymbol{A} = \boldsymbol{Q}^* \boldsymbol{D} \boldsymbol{Q}$ is a matrix that you know all the spectral information about.
   Fix $n = 2000$. Fixing the desired accuracy of GMRES at $1e-10$, solve $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ where $\boldsymbol{b}$ is vector with random entries and $\boldsymbol{A}$

   (a) has n distinct well separated spectrum.

   (b) is full rank but only has 3 distinct eigenvalues. 1 distinct eigenvalue.

   (c) is full rank and all eigenvalues live in a ball of radius $1e-5$ centered at 1.

   (d) is ill-conditioned. i.e. condition number $1e20$.

   (e) has one zero eigenvalue but $\boldsymbol{b} \in Range(\boldsymbol{A})$.

   What have you learned from these experiments? How does the number of required iteration change for each problem?
   Does this give you ideas of when GMRES performs well? Does it give you an idea of what you should be looking for in a preconditioner?

3 Software Expectations

It is expected that all of the code used in this project is your own except when the directions tell you otherwise or you have gotten approval to use a software package.

4 Independent Directions

Possible next steps for the project include, but are not limited to:

1. Investigating and reporting on different iterative methods for approximating eigenvalues.

2. Investigating and reporting on additional iterative solvers such as the Biconjugate Gradient Stabilized Method or multigrid.

3. Present how these methods are used in practice. Note: you will have to get professor approval to ensure your plan is mathematical enough.

References

[1] , G. Opfer and G. Schober Richardson's iteration for nonsymmetric matrices *Linear Algebra and its Applications*, 58: 343–361,1984.

[2] Y.Saad and M.H. Schultz GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems *SIAM Journal on Scientific and Statistical Computing*, 7(3): 856–869, 1986.