Arnoldi Iteration and Applications

Camille R. Zaug and Christopher Ross

March 15, 2019

1 Introduction and Motivation

The eigenvalues of a matrix or linear operator A give us important information about the system. They can help us construct an orthogonal basis that spans A (if possible), indicate how transformation acts on a vector, find approximate representations of A, and more [1]. They are therefore used in a wide variety of applications, from designing bridges to searching for oil [2]. A well-known method of finding the eigenvalues of a matrix is the QR decomposition, but this method is most effective for matrices on the order of 1000×1000 elements large [3]. In most real-world applications, the systems are much larger, so to find eigenvalues of these larger systems we turn to Arnoldi iteration.

Arnoldi Iteration is a relatively expedient way to approximate the largest eigenvalues of a square matrix A. This method can be thought of as a projection of A onto the order-n Krylov subspace $\mathcal{K}_n$ given by

$$\mathcal{K}_n(A, \mathbf{x}) = \text{span}\{\mathbf{x}, A\mathbf{x}, A^2\mathbf{x}, ..., A^{n-1}\mathbf{x}\}$$

where the vectors forming the basis of this subspace can then be used to find the eigenvalues of A [4]. In practice, we do not use the vectors $\{\mathbf{x}, A\mathbf{x}, A^2\mathbf{x}, ..., A^{n-1}\mathbf{x}\}$. As n increases, the vectors begin to approach an eigenvector pointing in the direction of the largest eigenvalue of A, resulting in many nearly-parallel vectors that form a poor basis for computational purposes. Instead, we choose a different method, namely the modified Gram-Schmidt method, to construct a basis for $\mathcal{K}_n$ [5].

The Arnoldi method is especially useful for studying the eigenvectors of linear operators, as they can do so without ever needing the matrix representation of that operator. This saves memory, since the matrix form is never created, and time, since the elements are not read off of the matrix.

2 The Arnoldi Iteration Algorithm

Given a square matrix A and a guess vector $\mathbf{b}$, the first step of the Arnoldi algorithm is to normalize $\mathbf{b}$ and use it as the first vector in the basis for $\mathcal{K}_n$, so

$$\mathbf{q}_0 = \frac{\mathbf{b}}{||\mathbf{b}||_2}.$$

The next n-$\mathbf{q}$ vectors are found by the first taking the product

$$\mathbf{q}_n = A\mathbf{q}_{n-1}$$

and then projecting out the components of $\mathbf{q}_n$ in the directions of the other $n-1$ $\mathbf{q}$ vectors. As a result, all vectors are orthogonal to one another. Finally, the vectors are normalized, and

$$\mathbf{q}_n = \frac{\mathbf{q}_n}{||\mathbf{q}_n||_2}.$$

The details of this method are given in algorithm 1.

Algorithm 1: Given a matrix A, a guess vector $\mathbf{b}$, and n iterations, The Arnoldi iteration algorithm returns an orthonormal matrix Q_n and a Hessenberg matrix H_n that can be used to find the eigenvalues of A. Note: $\mathbf{q}^H$ indicates the Hermitian of $\mathbf{q}$.

Result: Q_n, H_n
$\mathbf{q}_0 = \frac{\mathbf{b}}{||\mathbf{b}||_2}$
for *j = 1, 2, 3, ..., n* **do**
  $\mathbf{q}_j \leftarrow A(\mathbf{q}_{j-1})$
  for *i = 1, 2, ..., j+1* **do**
    $\mathrm{h}_{i,j-1} \leftarrow \mathbf{q}_i^H \mathbf{q}_j$
    $\mathbf{q}_j \leftarrow \mathbf{q}_j - h_{i,j-1}\mathbf{q}_j$
  end
  $\mathrm{h}_{j,j-1} \leftarrow \|\mathbf{q}_j\|$
  $\mathbf{q}_j \leftarrow \frac{\mathbf{q}_j}{h_{j,j-1}}$
end

There are a few restrictions on this method. First, $\mathbf{b}$ must not be the zero vector and it must not be an eigenvector of A. If so, performing the operation A$\mathbf{b}$ will simply result in a scaled version of $\mathbf{b}$ and thus an orthogonal basis cannot be formed [4]. Additionally, it may be desirable to ensure that $||\mathbf{q}_n||_2$ is not within a certain tolerance of zero, breaking out of the loop if this occurs. If $||\mathbf{q}_n||_2 = 0$, we have found an invariant subspace, and we will not be able to produce a vector $\mathbf{q}_{n+1}$ that is orthogonal to the other n vectors [5].

Performing the Arnoldi iteration n times results in two matrices, Q_n and H_n. Q_n is an orthonormal $n \times n+1$ matrix. Its columns are made up of the vectors $\mathbf{q}$, so Q_n is of the form

$$Q_n = \begin{bmatrix} | & | & | & \dots & | \\ \mathbf{q}_0 & \mathbf{q}_1 & \mathbf{q}_2 & \dots & \mathbf{q}_n \\ | & | & | & \dots & | \end{bmatrix}.$$

The values $h_{i,j}$ are elements of a $n+1 \times n$ upper Hessenberg matrix of the form

$$H_n = \begin{bmatrix} h_{1,1} & h_{1,1} & h_{1,1} & \dots & h_{1,n-1} & h_{1,n} \\ h_{2,1} & h_{2,1} & h_{2,1} & \dots & h_{2,n-1} & h_{2,n} \\ 0 & h_{3,1} & h_{3,1} & \dots & h_{3,n-1} & h_{3,n} \\ 0 & 0 & h_{4,1} & \dots & h_{4,n-1} & h_{4,n} \\ 0 & 0 & 0 & \ddots & h_{5,n-1} & h_{5,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & h_{n,n-2} & h_{n,n-1} & h_{n,n-2} \\ 0 & 0 & 0 & 0 & h_{n+1,n-1} & h_{n+1,n} \end{bmatrix}.$$

The Hermitian of a matrix Q is its complex conjugate transpose and denoted by Q^H. Given that a Hessenberg factorization of the matrix A is

$$A = Q^{\mathrm{H}} H Q,$$

where H is an upper Hessenberg and Q is orthonormal, we see that another interpretation of the Arnoldi iteration as the use of Gram-Schmidt orthogonalization in the Hessenberg factorization of A [6].

One major benefit of the Arnoldi iteration is that the iteration may be stopped at any time after n iterations and a basis for the subspace $\mathcal{K}_n$ will have been created and stored in Q_n, so even if we do not complete all iterations, we still receive useful information [6]. This method does have its downsides, as we still must store the (possibly large) vectors $\mathbf{q}$ in memory, and each iteration get more computationally expensive [5]. The Arnoldi iteration can be "restarted" to diminish this problem, using the output after a few iterations as a new input [8].

As a final note, if the matrix A is Hermitian (meaning that the complex conjugate of the lower diagonal elements are reflected across the upper half), the Arnoldi iteration simplifies into the Lanczos iteration [7]. This is simply an optimized version of the Arnoldi iteration for Hermitian matrices. The resulting matrix H_n is also Hermitian and tridiagonal, so only two vectors need be stored in memory; one for the diagonal and one for the subdiagonal. Additionally, to find the H_n, only the two most recent vectors $\mathbf{q}_n$ are needed since most values of H_n are zero. Thus, we see that the Lanczos iteration saves time and memory. Please see chapter 17 from J. Hupmpherys for a complete algorithm [4].

3 Finding Eigenvalues

We can express the matrix H_n as

$$H_n = Q_n^{\mathrm{H}} A Q_n,$$

The product $Q_n^{\mathrm{H}} A Q_n$ is a unitary transformation, which preserves the inner products of vectors [7]. In fact, H_n is a low-rank approximation of A. As n, the number of Arnoldi iterations, increases, the eigenvalues of H_n will converge to the largest eigenvalues of A. The eigenvalues of H_n are called the Ritz values [4].

3.1 Convergence

The Ritz values converge to the eigenvalues of A at different rates. Typically, the largest eigenvalues will converge before the smaller eigenvalues [9]. Figure 1, taken from J. Humphereys, shows that the largest eigenvalue for a random 500×500 matrix converges faster than the other 15 largest eigenvalues [4]. In addition, Ritz values for random matrices will take longer to converge than those for non-random matrices [4]. To decrease the number of iterations to converge, the Arnoldi method with restarting is often implemented instead, which reduces the number of iterative steps to achieve convergence [8].

3.2 Example: The Fast Fourier Transform

The Fast Fourier Transform is a computationally efficient way to compute the Fourier amplitudes of a signal, where Fourier amplitude associated with frequency k on a period L is defined by

$$\hat{f}(k) = \frac{1}{L} \int_0^L f(x) \exp(-2\pi i k x / L) dx.$$

Should we attempt to write this operator in matrix form to act on a complex vector u(x) with N elements, we would find

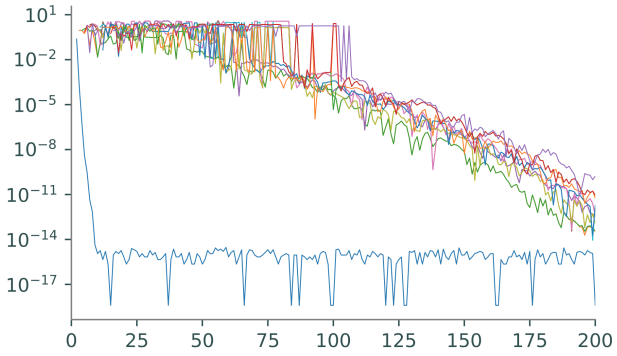

Figure 1: The convergence of 15 eigenvalues for a random matrix. The blue line is the largest eigenvalue, and it converges in 20 iterations [4].

$$\hat{\mathbf{u}}(x) = \begin{bmatrix} 1 & 1 & 1 & ... & 1 \\ \exp(-2\pi i k_1 x_1/L) & \exp(-2\pi i k_1 x_2/L) & \exp(-2\pi i k_1 x_3/L) & ... & \exp(-2\pi i k_1 x_N/L) \\ \exp(-2\pi i k_2 x_1/L) & \exp(-2\pi i k_2 x_2/L) & \exp(-2\pi i k_2 x_3/L) & ... & \exp(-2\pi i k_2 x_N/L) \\ \exp(-2\pi i k_3 x_1/L) & \exp(-2\pi i k_3 x_2/L) & \exp(-2\pi i k_3 x_3/L) & ... & \exp(-2\pi i k_3 x_N/L) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \exp(-2\pi i k_N x_1/L) & \exp(-2\pi i k_N x_2/L) & \exp(-2\pi i k_N x_3/L) & ... & \exp(-2\pi i k_N x_N/L) \end{bmatrix} \mathbf{u},$$

which is an $N \times N$ matrix [10]. If N is 2^{20}, we would need 64 terabytes of memory to store this matrix, so we cannot practically use the matrix form for the Fourier transform to find its eigenvalues [4].

Fortunately, Arnoldi iteration allows us to find the eigenvalues of a linear operator $\mathcal{L}$ (such as the FFT) without ever finding its explicit matrix representation. For a vector $\mathbf{u}$ N elements long, we can use Arnoldi iteration to find the four largest eigenvalues of FFT($\mathbf{u}$), which are known to be $\sqrt{N}, -\sqrt{N}, i\sqrt{N}$, and $-i\sqrt{N}$ [5]. Using the implementation of the Arnoldi iteration with a complex random guess vector $\mathbf{b}$, we were able to confirm this result for dimension 2^{20}. After only 4 iterations, we find the largest eigenvalues of the FFT acting on 2^{20} dimensions are indeed 1024, -1024, $1024i$, and $-1024i$, as expected. However, changing the guess vector changes the convergence of the Ritz values to the expected eigenvalues of A. For example, if the guess vector is a purely real vector of ones, we do not see the desired result.

4 GMRES

The Arnoldi iteration can be altered to solve a system of equations

$$\mathbf{A}x = \mathbf{b}.$$

The algorithm that solves this system is called the Generalized Minimal Residual Method, or GMRES [11]. Given a guess vector $\mathbf{b}$, GMRES uses a least squares approximation to obtain a vector $\mathbf{x}_n$ in the subspace $\mathcal{K}_n$ that minimizes $||A\mathbf{x}_n - \mathbf{b}||_2$, which is called the residual r_n [4]. The value of $\mathbf{x}_n$ formed after n Arnoldi iterations is given by

$$\mathbf{x}_n = Q_n \mathbf{y}_n + \mathbf{x}_0,$$

where $\mathbf{x}_0$ is an initializing guess vector and $\mathbf{y}_n$ is found by using least squares to minimizing the residual

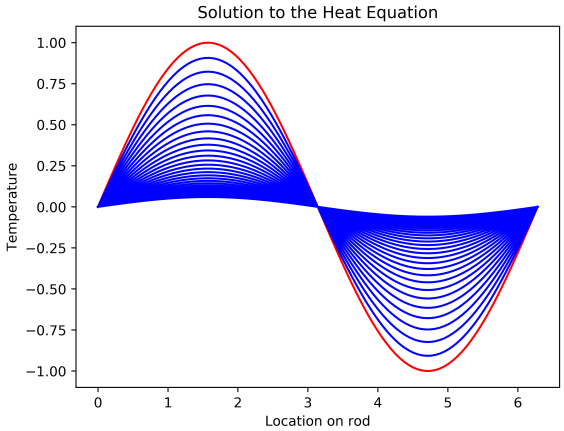

Figure 2: The solutions to the heat equation. The red curve is the initial temperature profile, and the blue curves, which decrease exponentially as time increases, represents the temperature profile of the rod at time n.

$$||H_n \mathbf{y}_n - (||\mathbf{b} - A\mathbf{x_0}||_2)\mathbf{e}_1||_2,$$

where $\mathbf{e}_1$ is a unit column vector $n+1$ units long. Here, Q_n and H_n are the same matrices returned by the plain Arnoldi iteration. Please see chapter 18 in J. Humpherys for a complete algorithm [4].

4.1 Convergence

The GMRES method converges linearly with the number of iterations, and each iteration takes more time and memory to compute. Therefore, this method can take prohibitively long to converge. As with the Arnoldi iteration, implementing GMRES with restarting can decrease the number of iterations necessary for convergence [9].

4.2 Example 1: The Forward-Time, Centered-Space Heat Equation

Given a one dimensional rod of length 2π, we can use GMRES numerically solve the heat equation

$$\begin{cases} u_t = u_{xx} \\ u(0,t) = 0 \\ u(2\pi),t) = 0 \\ u(x,0) = f(x) \end{cases}$$

with $u(x,t)$ representing the temperature of the rod at location x at time t and the initial temperature profile given by $f(x) = \sin(x)$.

We begin by discretizing $f(x)$ and u into N evenly spaced points with $\Delta x = (2\pi/N)$. Using forward-time finite and centered-space finite differences, we can write a matrix equation representing one step forward in time Δt,

$$\mathbf{u}_n = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ -\frac{\Delta t}{(\Delta x)^2} & 1+\frac{2\Delta t}{(\Delta x)^2} & -\frac{\Delta t}{(\Delta x)^2} & 0 & \dots & 0 \\ 0 & -\frac{\Delta t}{(\Delta x)^2} & 1+\frac{2\Delta t}{(\Delta x)^2} & -\frac{\Delta t}{(\Delta x)^2} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 \end{bmatrix} \mathbf{u}_{n-1},$$

where $\mathbf{u}_0$ will be $f(x)$. We use GMRES to solve this system of equations for the vectors $\mathbf{u}_n$, which represent the temperature profile of the rod at time n. For the conditions listed above and $\Delta t = 0.1$, these solution vectors are plotted for n=0,1,2...30 in figure 2. As expected, the temperature decays to zero as time increases. For this particular problem, GMRES requires only 1 iteration for each time n with a tolerance of 10^{-12}, indicating fast convergence.

5 Summary

The Arnoldi iterative method utilizes Krylov subspaces to find an approximate Hessenberg reduction of a square matrix A. It can also be modified to the Lanczos iteration for Hermitian matrices; which is more computationally efficient and requires less storage. The Arnoldi iteration is especially useful for finding the eigenvalues of a linear operator acting on a large vector without creating the matrix form of that operator. In addition, GMRES is an algorithm that solves systems of equations using the Arnoldi iteration. The Arnoldi iteration has its downfalls, mainly that each iteration takes more time and storage than the last, but this issue can be reduced via restarting. Generally, this algorithm and its adaptations are very effective at studying large matrices.

References

[1] Where do eigenvalues/eigenvectors/eigenfunctions come from, and why are they important anyway? `http://www.math.tamu.edu/~mpilant/math311/eigenvalues.pdf`.

[2] Some Applications of the Eigenvalues and Eigenvectors of a Square Matrix: `https://www.cpp.edu/~manasab/eigenvalue.pdf`.

[3] E. Jarlebring, Introduction to Arnoldi Method: `https://www.math.kth.se/na/SF2524/matber14/arnoldi_intro_pub.pdf`, (2007).

[4] J. Humpherys and T. J. Jarvis, Labs for Foundations of Applied Mathematics: Volume 1 Mathematical Analysis.

[5] Arnoldi and Lanczos Algorithms: `http://people.inf.ethz.ch/arbenz/ewp/Lnotes/chapter10.pdf`.

[6] Arnoldi Iteration and GMRES: `http://www.math.iit.edu/~fass/477577_Chapter_14.pdf`.

[7] Finding Eigenvalues: W. Wheeler, Arnoldi Iteration and the QR Algorithm: `http://algorithm-interest-group.me/assets/slides/Eigenvalue_algorithms.pdf`, (2016).

[8] Restarting Arnoldi and Lanczos Algorithms: `http://people.inf.ethz.ch/arbenz/ewp/Lnotes/chapter11.pdf`.

[9] M. Bellalik, Y. Saad, and H. Sadok, On the Convergence of the Arnoldi Process for Eigenvalue Problems: `https://www-users.cs.umn.edu/~saad/PDF/umsi-2007-12.pdf`.

[10] B. W. Dickinson and K. Steiglitz, Eigenvectors and Functions of the Discrete Fourier Transform: `http://www.cs.princeton.edu/~ken/Eigenvectors82.pdf`, (1982).

[11] Generalized Minimal Residual Method: `http://mathworld.wolfram.com/GeneralizedMinimalResidualMethod.html`.