Eigensystems for Large Matrices

L. Harris
I. Papanikolaou
J. Shi
D. Strott
Y. Tan
C. Zhong

Dr. Changhui Tan
AMSC460 - Spring 2015

May 12, 2015

Abstract

In this paper we will introduce the ideas behind several methods that have been developed to solve Eigensystems for Large matrices. These methods include the Power Iteration Method, the QR Method, the Jacobi Method, but we will also briefly discuss some of their variants and other important approaches, continuously presenting the reader with the motivating factors behind each of the methods, their advantages but also their limitations.

Contents

1 Introduction

In Mathematics, the theory of eigenvalues and eigenvectors of matrices are extremely significant for solving many Engineering, Physics, Computer Science and a plethora of real-life problems. To obtain solutions for these eigensystems, we are interested in finding the roots of $det(A - \lambda) = 0$, which is called the **characteristic polyonomial**, where A is a matrix of dimensions $n \times n$ and λ the eigenvalues. For small n, this can easily be computed by hand, but for large n, the computations are extremely costly, as we would have to find the roots of the **characteristic polynomial** of degree n.

It's very challenging and costly to find the solution of a the **characteristic polynomial** of degree n as n gets larger. However, for most applications we may not necessarily be interested in all the roots of the polynomial, but rather, only its largest (in absolute value) root; its largest eigenvalue, the second largest root; second largest eignevalue, or the smallest root; smallest eigenvalue.

In order to do this, we will present the following methods in the corresponding order: Power Iteration Method, QR Method, Jacobi Method while also presenting several variants and mentioning other important methods.

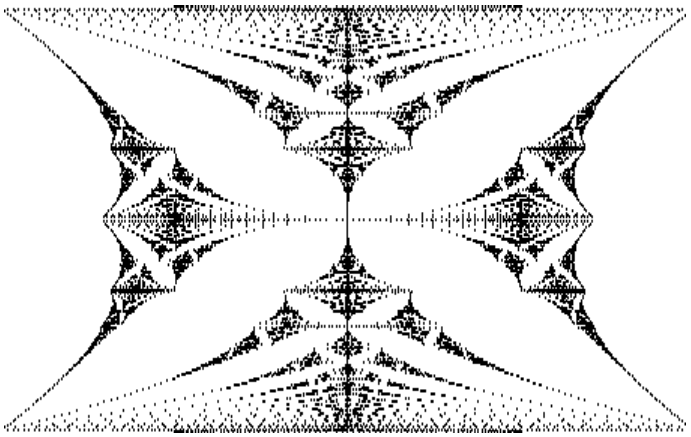

Figure 1: Energy level spectrum for electrons in Z^2 as a function of magnetic field strength (vertical coordinate ranging from 0 to 2π, horizontal from -4 to 4). This is the eigenvalue spectrum for n large of the tridiagonal n $\times$ n matrix whose entries one above and below the diagonal are 1, and diagonal entries are 2, 2Cos(t),2Cos(2t),... where t is the vertical coordinate. (*http://www.math.brown.edu/ rkenyon/gallery/gallery.html*)

2 Power Iteration

In the Power Iteration method, we are interested in obtaining the largest eigenvector, from which we can extract the largest eigenvalue using the Rayleigh Quotient (will be discussed in the last section.

Similar to other algorithms, such as the Jacobi and Gauss-Seidel methods, our tactic here is iteration. Power iteration, also known as the Von Mises iteration, results in finding the eigenvector corresponding to the dominant eigenvalue. We may also use the inverse of this method in order to find the eigenvector associated with the smallest eigenvalue. Many pros and cons arise with this method. Most importantly, it is very simplistic due to the lack of matrix decomposition. This means the Power Method is very efficient - almost instantaneous - and we can use large, sparse matrices. But, here we only find the single, dominant eigenvalues; finding more than one eigenvalue leads to a variation of the method.

2.1 The Method

Assumptions:

- Matrix A is diagonalizable/linerly independent eigenvectors, $x_1, x_2, ..., x_n$, such that $Ax_i = \lambda_i x_i$ for $i = 1, ..., n$.
- Matrix A obtains a dominant eigenvalue, say λ_1, such that $|\lambda_1| > |\lambda_2| \geq ... \geq |\lambda_n\|$

We begin with a random vector x_0 and create a sequence as follows :
$x_{i+1} = \frac{Ax_i}{||Ax_i||_\infty}$ where our eigenvalue is $\lambda_{i+1} = ||Ax_i||_\infty$.
Our iteration reads:

1. Let k=0 and obtain an initial vector $x^{(0)}$ where $||x^0|| = 1$.
2. k := k+1
3. $y^{(k)} := Ax^{k-1}$;
4. $x^{(k)} := \frac{y^{(k)}}{||y^{(k)}||_\infty}$
5. Repeat steps (2) through (4) until proper convergence is found.

The proof of this method can be found in John H. Mathews's Power Method on page 603. [1]

2.2 Convergence Rate

The convergence (if existent) is determined by $\frac{|\lambda_2|}{|\lambda 1|}$. Thus, if the next dominant eigenvalue is close in value, the iteration will converge slowly. We also notice failure to converge if λ_1=λ_2 OR $|\lambda_1|$=$|\lambda_2|$ with $\lambda_1 \neq \lambda_2$ (same value, with opposite signs).

The details can be found on Larson's textbook p.555 [2]

2.3 Numerical Example

Let A be the matrix, $A = \begin{bmatrix} 2 & -12 \\ 1 & -5 \end{bmatrix}$ which has a dominant eigenvalue.

Using six iterations, we will compute the dominant eigenvector.

We begin with an approximation vector x_0, $x_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

Our iterations are as follows:

$$x_1 = Ax_0 = \begin{bmatrix} 2 & -12 \\ 1 & -5 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -10 \\ -4 \end{bmatrix} \longleftrightarrow -4 \begin{bmatrix} 2.50 \\ 1.00 \end{bmatrix}$$

$$x_2 = Ax_1 = \begin{bmatrix} 2 & -12 \\ 1 & -5 \end{bmatrix} \begin{bmatrix} -10 \\ 4 \end{bmatrix} = \begin{bmatrix} 28 \\ 10 \end{bmatrix} \longleftrightarrow 10 \begin{bmatrix} 2.80 \\ 1.00 \end{bmatrix}$$

$$x_3 = Ax_2 = \begin{bmatrix} 2 & -12 \\ 1 & -5 \end{bmatrix} \begin{bmatrix} 28 \\ 10 \end{bmatrix} = \begin{bmatrix} -64 \\ -22 \end{bmatrix} \longleftrightarrow -22 \begin{bmatrix} 2.91 \\ 1.00 \end{bmatrix}$$

$$x_4 = Ax_3 = \begin{bmatrix} 2 & -12 \\ 1 & -5 \end{bmatrix} \begin{bmatrix} -64 \\ -22 \end{bmatrix} = \begin{bmatrix} 136 \\ 46 \end{bmatrix} \longleftrightarrow 46 \begin{bmatrix} 2.96 \\ 1.00 \end{bmatrix}$$

$$x_5 = Ax_4 = \begin{bmatrix} 2 & -12 \\ 1 & -5 \end{bmatrix} \begin{bmatrix} 136 \\ 46 \end{bmatrix} = \begin{bmatrix} -280 \\ -94 \end{bmatrix} \longleftrightarrow -94 \begin{bmatrix} 2.98 \\ 1.00 \end{bmatrix}$$

$$x_6 = Ax_5 = \begin{bmatrix} 2 & -12 \\ 1 & -5 \end{bmatrix} \begin{bmatrix} -280 \\ -94 \end{bmatrix} = \begin{bmatrix} 568 \\ 190 \end{bmatrix} \longleftrightarrow 190 \begin{bmatrix} 2.99 \\ 1.00 \end{bmatrix}$$

So we can see that using six iterations, our vector approaches $\begin{bmatrix} 3 \\ 1 \end{bmatrix}$, which is the dominant eigenvector of the system.

2.4 Pseudocode: Power Iteration Algorithm

The pseudocode for the Power Iteration method is[3]:

---

1. for $k = 1, 2, 3...$
2. $w = A * v$
3. $\lambda = norm(w)$
4. $A_i = Q_i R_i$
5. $v = w/\lambda$
6. end

2.5 Variant: Inverse Iteration

Another significant goal behind this algorithm is to find the smallest eigenvalue, in which we turn to the Inverse Power Iteration. We will return the eigenvector corresponding to the intended eigenvalue; this method works inversely to the original scheme (hence, the name). If the eigenvalues are so small, the convergence becomes very slow. So, we will use a shifted matrix $A - \sigma I$ and obtain the eigen pair (μ_i, u_i) with $\mu = \frac{1}{\sigma - \lambda_i}$. Our iteration reads:

1. Choose an initial vector x_0 and a shift σ.
2. Compute the LU factorization of $A(\sigma)I : LU = P(A(\sigma)I)$
3. $y^{(0)} := U^{-1}L^{-1}Px^{(0)}$, $\mu^{(0)} = y^{(0)*}$, $\lambda^{(0)} := \sigma + \frac{1}{\mu^{(0)}}$; $k := 0$.
4. Let $k := k + 1$
5. $x^{(k)} := \frac{y_{k-1}}{||y_{k-1}||}$.
6. $y^{(k)} := U^{-1}L^{-1}Px^{(k)}$.
7. $\mu^{(k)} := y^{(k)^*}x^{(k)}$; $\lambda^{(k)} := \sigma + \frac{1}{\mu^{(k)}}$.
8. Repeat steps (4) through (8) until we hit the convergence criterion, where: $||x^{(k)} - \frac{y^{(k)}}{\mu^{(k)}}|| \leq tol||y^{(k)}||$.

The proof of this method can be found in John H. Mathews's Power Method on page 605.[4]

2.6 Real Life Applications

Besides Engineering, Physics, and Computer Science, the power method is implemented in many real life scenarios which are often over looked. For example, Twitter uses power iteration in suggesting followers by ranking popularity. More importantly, Google is ran by this method; PageRank, also known as the $25,000,000,000 eigenvector [5], is essentially what defines the search engine. It was invented in 1998 by Sergey Brin and Larry Page. Search engines, in general, have the following jobs:

1. Locating all accessible web pages which will be ranked.
2. Indexing this data to be easily searched for via main key words.
3. Rating the importance of each page in the database so that most relevant and accurate results arise for the web user.

The third step is where PageRank becomes crucial; Google was able to discover this algorithm and deliver much more sufficiently than the other search engines. Finding the dominant eigenpair works quickly and in parallel to finding the most important links; the link with the most hits becomes the most dominant page. The success of these applications derives from the efficiency behind the method.[6]

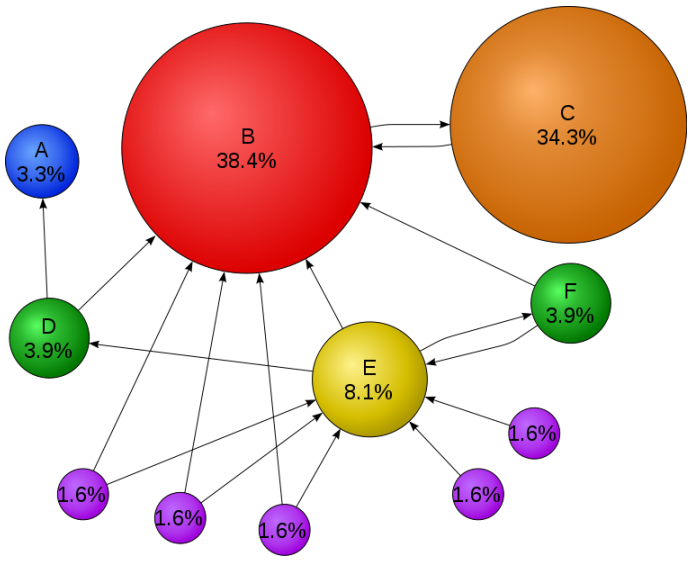

Figure 2: PageRank for a simple network (*http://en.wikipedia.org/wiki/PageRank*)

Furthermore, another significant application is how Facebook uses this method to calculate the relative level of importance of a person's friends and obtain a weight measurement that contributes as a factor in how the newsfeed items get displayed for their friends. To measure influence, Facebook uses clever algorithms and graph theory to derive a large matrix with the each user's "influence factor". Then, using techniques such as the Power Iteration Method, it solves these large matrices to find the most influential people in a cluster. More details can be found on Jesse Farmer's webpage [7].

3 The QR Algorithm

The power iteration method solved the issue of locating the dominant eigenvalue and eigenvector. It can also be good especially for sparse matrices because it does not compute a full matrix decomposition. However, there are times when we need to compute all of the eigenvalues and eigenvectors for a particular matrix and this matrix may be very large and dense. This is where the QR algorithm comes into play. While the QR algorithm can work for both real and complex matrices, we focused mainly on real matrices for our research. The idea behind this algorithm is to start with a non-singular matrix A and apply an iterative procedure to convert A into similar matrix whose eigenvalues are easy to extract. In its most basic form the QR algorithm is as follows:

- $A_0 = A$
- $A_0 = Q_1 R_1$
- $A_k = R_k Q_k, k \geq 1$

The QR in the above algorithm are the same as in the QR factorization. The idea is that we decompose the starting matrix into an orthonormal matrix Q and an upper triangular matrix R. Then to get the matrix for the next iteration we reverse the order and multiply RQ. It is easy to see that since Q is orthonormal, $Q^T A = R$ and $A_k = Q_k^T A_{k-1} Q_k$. Thus, At any step in the process, A_k is similar to the original matrix A. So, if this process produces a matrix whose eigenvalues are easily obtained, then we have the eigenvalues of the original matrix. Luckily for us, the QR algorithm is doing just this by producing a schur decomposition of the starting matrix. The schur decomposition means finding a matrix U and a matrix T such that $T = U^* A U$. Where T is upper triangular with the eigenvalues of A on its diagonal and U is unitary[9]. Now, in the case where A is complex, this decomposition is guaranteed to exist (see Theorem 2.8 of Arbenz)[9]. In the case of A being real, there exists a Q and a R such that $R = Q^T A Q$ with Q orthogonal and R upper triangular with the eigenvalues of A on its diagonal if all of the eigenvalues of A are real. If A has complex eigenvalues, R will be upper-quasi triangular, see Theorem 2.10 of Arbenz; however, we didn't treat this case and the methods we considered had issues with convergence if A had any complex eigenvalues[9].

3.1 Limitations and Improvements

The main issues with the algorithm in its basic form have to deal with the following:

1. Computational cost of each iteration
2. Speed of convergence (if it converges)

The first issue is handled by finding a matrix structure that is well suited to the QR decomposition and that is preserved under the $Q^T A Q$ transformation. While upper triangular would be best, not all matrices are similar to an upper triangular matrix. However, all matrices are similar to a Hessenberg matrix which has all zero below the first subdiagonal (upper

hessenberg) or all zero above the first superdiagonal (lower hessenberg) or both (tridiagonal, which will be the case if the real matrix is symmetric). Theorem 3.2 of Arbenz provides a proof that the Hessenberg form is preserved by the QR algorithm[9]. This is important because we only have perform a finite number of initial steps to transform the original matrix into a form that will reduce the cost of each step in the QR algorithm without any need to fix the matrix between each iteration.

The process of reducing a matrix to hessenberg form involves a sequence of similarity transformations. This is usually accomplished using a Householder reduction. This works by multiplying the matrix A by a unitary matrix P that will zero out all the elements below the element in the first sub-diagonal and the multiplying by the same matrix P to achieve similarity. This process is repeated with a different P at each stage to achieve hessenberg form. A more detailed explanation of this process can be found in section 3.3 of Arbenz[9].

Now that we have converted the starting matrix to Hessenberg form, we are in a better place in terms of computational complexity. A QR step with a full matrix costs $\frac{7}{3}n^3$ whereas with Hessenberg form only costs $12n^2$ giving us a gain of an order of n (further reading on complexity can be found on Wikipedia as well as section 3.2.2 of Arbenz)[9]. Now that we have reduced the computational complexity, the issue of convergence still remains. As mentioned previously, we will have issues if the matrix has complex eigenvalues, so for the rest of this discussion we will assume that it has real eigenvalues. Even with all real eigenvalues we run into some issues. These issues stem from the fact that eigenvalues can have multiplicity greater than one, meaning some of the eigenvalues might be identical and from the fact that the eigenvalues even if all distinct, might be relatively close together in magnitude. However, if we look at the cases where the algorithm does converge the convergence could be slow. The way to address this and to potentially help convergence in the case where eigenvalues are close in magnitude is to introduce a shift for the matrix into the algorithm.

The motivation for the shift comes from looking at an irreducible hessenberg matrix H and performing a QR decomposition on said matrix. The result is that for R, $|r_{kk}| > 0, k < n$[9]. So H will be singular if $r_{nn} = 0$ (for proof see Lemma 3.4, Arbenz)[9]. If we look at the decomposition of the matrix $H - \lambda I = QR$ and consider the previous statement, R will have $r_{nn} = 0$ since λ is an eigenvalue of H. So if we then let our next iterate equal $H + \lambda I$ the eigenvalue will be in the bottom right diagonal entry with zeroes to its left. Essentially, an eigenvalue has been removed from H and placed in the bottom right diagonal entry of a similar matrix, which we can now deflate and repeat our algorithm with a smaller similar matrix (page 62 of Arbenz provides a nice visual of the process)[9]. The problem is that we don't actually know the eigenvalues of H, they are what we are trying to find. However, the procedure just illustrated above indicates that if we pick a value close to an eigenvalue and perform the shift, the bottom right diagonal entry should converge towards an eigenvalue and the entry to the left of it should get very small. Once the element to the left of it is sufficiently small, we can declare the (n, n-1) entry zero and take the (n,n) entry to be our eigenvalue. We can then deflate the matrix again as above and repeat the procedure. A common choice for the shift is Rayleigh quotient shift, which uses the last diagonal element of H in the first step and then uses the last diagonal element of each subsequent H in the algorithm at each step.

In practice there are many other types of shifts used. What type of shift to use depends on the specific matrix and what issues with convergence are being observed. The single shift

mentioned above works for matrices with real eigenvalues. If there are complex eigenvalues involved, a double-shift should be performed which helps find pairs of eigenvalues that occur when dealing with complex conjugates.

3.2 Remarks

As mentioned at the beginning the QR algorithm is best suited for full dense matrices. It is ill-advised to use the algorithm on sparse matrices as the reduction to Hessenberg form will generate fill ins for most if not all of the empty space in the sparse matrix destroying its original structure. There are better algorithms for dealing with sparse matrices. However, if the goal is to find all the eigenvalues of a large dense matrix, the QR algorithm especially when modified with shifting is extremely powerful.

3.3 Pseudocode: QR Algorithm

The pseudocode for the QR method is[10]:

---

1. $i = 0$
2. $A_1 = A$
3. repeat
4. Factor $A_i = Q_i R_i$
5. $A_{i+1} = R_i Q_i = Q_i^T A_i Q_i$
6. $i = i + 1$
7. until convergence

---

4 Jacobi Eigenvalue Method

4.1 Rotation Matrix in $\mathbb{R}^2$

The rotation matrix in $\mathbb{R}^2$ is defined by $R = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix}$, which will rotate the basis in $\mathbb{R}^2$ counterclockwisely by angle ϕ. The inverse of rotation matrix, similarly, is given by $R^{-1} = \begin{bmatrix} \cos(-\phi) & \sin(-\phi) \\ -\sin(-\phi) & \cos(-\phi) \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$. It rotates the basis clockwisely.

4.2 The 2×2 Symmetric Schur Decomposition

Let S be a symmetric matrix.

$$R^{-1}SR = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix}^T \begin{bmatrix} a_{pp} & a_{pq} \\ a_{pq} & a_{qq} \end{bmatrix} \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} = \begin{bmatrix} b_{pp} & 0 \\ 0 & b_{qq} \end{bmatrix}$$

Therefore, the angle ϕ satisfies equation:

$$\cos\phi = \frac{1}{\sqrt{1+\tau^2}}, \quad \text{where} \quad \tau = \frac{a_{pp} - a_{qq}}{2a_{pq}} \tag{1}$$

or

$$\tan 2\phi = \frac{2a_{pq}}{a_{qq} - a_{pp}} \tag{2}$$

These equations are not obvious, but we can solve one of them and obtain a proper angle between $-\frac{\pi}{4}$ to $\frac{\pi}{4}$ to diagonalize our 2 by 2 matrices. [12]

4.3 Rotation Matrix in $\mathbb{R}^n$

The rotation operator in $\mathbb{R}^n$ is a generalization of 2 by 2 case.

$$R_{pq} = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & 0 & \\ & & \cos\phi & \cdots & \sin\phi & & \\ & & \vdots & \ddots & \vdots & & \\ & & -\sin\phi & \cdots & \cos\phi & & \\ & 0 & & & & \ddots & \\ & & & & & & 1 \end{bmatrix}; \quad R_{pq}^{-1} = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & 0 & \\ & & \cos\phi & \cdots & \sin\phi & & \\ & & \vdots & \ddots & \vdots & & \\ & & -\sin\phi & \cdots & \cos\phi & & \\ & 0 & & & & \ddots & \\ & & & & & & 1 \end{bmatrix}$$

We operate the transformation like we did for 2 by 2 rotation: conjugating the rotation matrix on original symmetric matrix S.

$$S' = R_{pq}^{-1} \cdot S \cdot R_{pq} = R_{pq}^{-1} \cdot \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & \ast & \\ & & a_{pp} & \cdots & a_{pq} & & \\ & & \vdots & \ddots & \vdots & & \\ & & a_{pq} & \cdots & a_{qq} & & \\ & \ast & & & & \ddots & \\ & & & & & & 1 \end{bmatrix} \cdot R_{pq} = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & \ast & \\ & & b_{pp} & \cdots & 0 & & \\ & & \vdots & \ddots & \vdots & & \\ & & 0 & \cdots & b_{qq} & & \\ & \ast & & & & \ddots & \\ & & & & & & 1 \end{bmatrix}$$

By choosing a non-zero and off-diagonal element and by the rotation transformation defined above, we can find a similar matrix S', such that the target off-diagonal element reduced to zero. [13]

4.4 Jacobi Eigenvalue Algorithm

Iteration

To diagonalize an arbitrary symmetric matrix, we can apply the rotation in $\mathbb{R}^n$ for every off-diagonal element. We can always start from the largest off-diagonal element in absolute value $|a_{pq}|_{max}$, and iterate:

$$S_{(1)} = R_{(1)}^{-1} S_{(0)} R_{(1)}, \quad \text{where } R_{(1)} \text{ related to } |a_{pq}^{(1)}|_{max}, \quad \text{and } S_{(1)} \sim S_{(0)}. \tag{3}$$

$$S_{(k+1)} = R_{(k+1)}^{-1} S_{(k)} R_{(k+1)}, \quad \text{where } R_{(k+1)} \text{ related to } |a_{pq}^{(k+1)}|_{max}, \quad \text{and } S_{(k+1)} \sim S_{(k)}. \tag{4}$$

The matrix $S^{(K)}$ will become more and more sparse, and eventually off-diagonal part can be reduced to zero.

Convergence

There are exact $N = \frac{n(n-1)}{2}$ off-diagonal elements we need to deal with. From the reference, the sequence of sweeps converges at least linearly with a factor $\approx e^{1/2}$, and the convergence can be quadratic if the matrix is under some additional conditions.

Computation Cost

A number of N Jacobi rotations is called a sweep. Each rotation has $O(n)$ cost and one sweep has $O(n^3)$ cost which is equivalent to one matrix multiplication.[14]

4.5 Generalizations and Applications

Hermitian Matrix

Hermitian Matrix is defined by,

$$H^\dagger = H \tag{5}$$

$H^\dagger$ is transposed complex conjugate of original complex-valued matrix H.
Moreover, let's define the rotation matrix R (satisfing unitary argument: $R^\dagger = R^{-1}$),

$$(R_{pq})_{mn} = \delta_{(m,n)(p,q)}; \tag{6}$$

$$(R_{pq})_{pp} = \frac{1}{\sqrt{2}}e^{-i\theta}; (R_{pq})_{qp} = \frac{1}{\sqrt{2}}e^{-i\theta}; (R_{pq})_{pq} = \frac{-1}{\sqrt{2}}e^{i\theta}; (R_{pq})_{qq} = \frac{1}{\sqrt{2}}e^{i\theta}; \tag{7}$$

The Jacobi method can be generalized to complex hermitian matrices by using a different rotation R but similar process. It will still preserve $O(n^3)$ efficiency. Solving eigenvalues for hermitian matrices has widely used applications.

In quantum physics, when people studying the time independent Schroedinger equations, eigenvalues of total energy operator Hamiltonian, represented by an hermitian matrix in Hilbert Space, are energy levels of the system for corresponding quantum states.

People will need to diagonalize the matrix of Hamiltonian, and find its eigenvalues. By generalized Jacobi Method, the energy level problem can be solved numerically. The iteration can give all energies for all quantum states. And this method can be much more efficient than finding analytic solutions especially for those complicated and unsolvable systems.

Parallelism

The Jacobi Method is also well suited for parallelism, which will not be applied in QR method. This greatly help us to boost the speed of finding eigenvalues by computer.[14]

4.6 Pseudocode: Jacobi Algorithm

The pseudocode for the Jacobi method is:

---

1. Build $\mathbf{A}, \mathbf{b}$
2. Build modified $\mathbf{A}$ with diagonal zero $\rightarrow \mathbf{Q}$
3. Set initial guess $\mathbf{x} = \mathbf{0}$
4. Do {
5. a) Compute $\tilde{x_i} = \frac{1}{\mathbf{A_i}} \left(b_i - \sum_{j=1}^{j=N} \mathbf{Q_{ij}} x_j \right)$
6. b) Compute Error, $err = \frac{\underset{i=1,\ldots,N}{max}(|x_i - \tilde{x_i}|)}{\underset{i=1,\ldots,N}{max}(|b_i|)}$
7. c) Update x, $x = \tilde{x_i}$
8. } while $err > tol$

---

5 Divide and Conquer Method

5.1 The Idea

This method is for Hermitian or real symmetric matrix. It can break the large matrix into smaller two pieces (or more). These smaller portions can be solved easier and faster than the original large matrix. Than the original problem can be solved by combing the two smaller parts together.[6]

5.2 Process

1. Reduce the matrix to tridiagonal.
2. Divide the matrix into two or more smaller parts.

$$T = \begin{bmatrix} \hat{T}_1 & 0 \\ 0 & \hat{T}_2 \end{bmatrix} + \begin{bmatrix} & \\ \beta & \beta \\ \beta & \beta \end{bmatrix}$$

3. Solve the smaller matrix

   $\hat{T}_1 = Q_1 D_1 Q_1^T$

   $\hat{T}_2 = Q_2 D_2 Q_2^T$

4. Combine the small parts to solve the original matrix.

   $$T = \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix} \left(\begin{bmatrix} D_1 & \\ & D_2 \end{bmatrix} + \beta z z^T \right) \begin{bmatrix} Q_1^T & \\ & Q_2^T \end{bmatrix}$$

5. Now the solution of the original matrix is given by the combination of diagonal matrix D_1 D_2 with a rank 1 shift from $\beta * z * z^T$, let $w * w^T = \beta * z * z^T$, then the eigenvalues and eigenvectors are given as the following:

   Eigenvalue: $1 + \sum_{j=1}^{m} \frac{w_j^2}{d_j - \lambda} = 0$

   Eigenvector: $(D + ww^T)q = \lambda q$

5.3 Advantages

1. This method is about 2 times faster when finding an eigenvector of N by N symmetric matrix. The step size of regular QR Methods are $6 * N^3$, while the divide and conquer method takes about $4/3 * N^3$.[7]
2. This method can also save some space during the process. Instead of saving the whole matrix, it can save space by just storing the smaller matrix and the remaining constants.

6 Lanczos and Arnoldi Methods

6.1 The Idea

During power iteration, there are many calculation used in the process, but only the last entry is used to get the largest eigenvalue. This is kinda of a waste of effort. We can use the unused terms and find the eigenvalues.

There is something called Krylov Subspace formed in the process of power iteration, so Lanczos method and Arnoldi method both use iteration methods to find orthonormal basis for the Krylov subspace, and transform the original matrix to something that is easy to solve, and these matrix are similar to the original one. So these transformed matrix will provide good approximations of the eigenvalues of the original matrix.[8][9][10]

6.2 Process

There are many methods in transforming the original matrix. The final result for Arnoldi method is a upper Hessenberg Matrix, and the result for Lanczos method is a tridiagonal matrix. Both of these types of matrix are easy to solve using QR or some other methods.[9][10]

$$\text{Arnoldi:} \quad H_n = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & \cdots & h_{1,n} \\ h_{2,1} & h_{2,2} & h_{2,3} & \cdots & h_{2,n} \\ 0 & h_{3,2} & h_{3,3} & \cdots & h_{3,n} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_{n,n-1} & h_{n,n} \end{bmatrix}$$

$$\text{Lanczos: } T_{mm} = \begin{bmatrix} \alpha_1 & \beta_2 & & & & 0 \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \alpha_3 & \ddots & & \\ & & \ddots & \ddots & \beta_{m-1} & \\ & & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ 0 & & & & \beta_m & \alpha_m \end{bmatrix}$$

So after we solve those matrix, the achieved eigenvalues are good approximations of the original matrix.

6.3 Advantages and Disadvantages

1. Arnoldi Method

- The process is stable
- The process produces a Hessenberg matrix, which can be solved efficiently
- But some version of this process can only produce one eigenvalue at a time, so the whole process could be time consuming. [7]

2. Lanczos Method

- It can produce a tridiagonal matrix, which can be stored in space efficiency manner and can be solved easily.
- This method has low linear complexity.
- The disadvantage is that sometimes the iteration process will produce a basis that is nor orthonormal, which will take many efforts to correct. [10]

7 Rayleigh Quotient

7.1 The Idea

The Idea behind the Rayleigh Quotient is rather simple. Instead of the normal way in finding eigenvalue first and then the corresponding eigenvector, this method allow people to use the eigenvector to find the corresponding eigenvalue. So if we start with a close approximation of the eigenvector, than we can have a good approximation of the eigenvalue.

7.2 Process

The process is rather simple. First lets see what Rayleigh Quotient
Given a vector $x \in R^n$ and a matrix A $\in R^(n * n)$, the associated *Rayleigh quotient* $R(x)$ is defined as the real number:

$$R(M, x) := \frac{x^* M x}{x^* x}$$

Also we can rewrite the equation to the following form:

$$R(M, x) = \frac{x^* M x}{x^* x} = \frac{\sum_{j=1}^{n} \lambda_i y_i^2}{\sum_{j=1}^{n} y_i^2}$$

Where (λ , v_i) is the eigenvalue and its corresponding eigenvector. And the $y_i = v_i * x$. So clearly if we have a vector x that is one of the eigenvectors, the result of the right hand side will just be the corresponding eigenvalue.

If we have a good approximation of the eigenvector, then we can have the corresponding eigenvalue, with some error. This provides a good way to estimate the eigenvalue. [11]

7.3 Advantages and Disadvantages

1. Fast convergence. The convergence rate is quadratic in general, and cubic in the case of Hermitian matrix.
2. This method has high iteration complexity. [11]

8 Conclusion

In this report, we have presented some real-life applications of eigenvalue and eigenvector, and we have gone through many methods for solving eigensystems for large matrices. Every method has its own unique advantages, disadvantages and specifications. Finding the most suited method for different situations is very critical in order to obtain the desired results. I hope after reading this report, we've familiarized the reader with some important terminology and a basic understanding of the individual methods.

Additionally, if the reader is interested in the corresponding presentation slides relating to this report, where several valuable (summarized) information can be found along with some valuable applications, we direct you to this group's presentation. [21]

Group Information

Power Iteration Method:

Laura Harris
Undergraduate Student (Senior)
Department of Mathematics
University of Maryland, College Park

Iason Papanikolaou
Undergraduate Student (Senior)
Department of Mathematics
University of Maryland, College Park

QR Method:

David Strott
Undergraduate Student (Senior)
Department of Mathematics
University of Maryland, College Park

Jacobi Method:

Junyi Shi
Undergraduate Student (Senior)
Department of Mathematics
University of Maryland, College Park

Yuewen Tan
Undergraduate Student (Senior)
Department of Physics
University of Maryland, College Park

Other Methods:

Chen Zhong
Undergraduate Student (Senior)
Department of Mathematics
University of Maryland, College Park

References

[1] John H. Mathews *Power Method*
http://mathfaculty.fullerton.edu/mathews//n2003/powermethod/PowerMethodProof.pdf

[2] Ron Larson *Elementary Linear Algebra*
ISBN-13: 978-1133110873

[3] Lothar Reichel *Lecture 14: Eigenvalue Computations*
http://www.math.kent.edu/ reichel/courses/intr.num.comp.2/lecture21/evmeth.pdf

[4] John H. Mathews *The Power Method for Eigenvectors*
http://mathfaculty.fullerton.edu/mathews/n2003/PowerMethodMod.html

[5] Kurt Bryan and Tanya Leise
THE 25,000,000,000 EIGENVECTOR THE LINEAR ALGEBRA BEHIND GOOGLE
http://www.rose-hulman.edu/ bryan/googleFinalVersionFixed.pdf

[6] Marziah Karch *What Is PageRank and How Do I Use It?*
http://google.about.com/od/searchengineoptimization/a/pagerankexplain.htm

[7] Jesse Farmer *Graph Theory: Part III (Facebook)*
http://20bits.com/article/graph-theory-part-iii-facebook

[8] Jim Lambers *The Eigenvalue Problem: Power Iteration*
http://www.math.usm.edu/lambers/mat610/sum10/lecture14.pdf

[9] Arbenz, Dr. Peter, Dr. Daniel Kressner,
Lecture Notes on Solving Large Scale Eigenvalue Problems,
http://people.inf.ethz.ch/arbenz/ewp/Lnotes/lsevp.pdf, Spring 2012.

[10] John H. Mathews - California State Univ. Fullerton *The QR Method for Eigenvalues*
http://mathfaculty.fullerton.edu/mathews/n2003/QRMethodMod.html

[11] Endre Süli, David F. Mayers, *An Introduction to Numerical Analysis*,
Cambridge University Press, 2003.

[12] Jim Lambers, *CME Lecture 7 Notes*, http://web.stanford.edu/class/cme335/lecture7.pdf

[13] Online Source, *Jacobi Rotation*, http://en.wikipedia.org/wiki/Jacobi_rotation

[14] Online Source, *Jacobi Eigenvalue Method*, http://en.wikipedia.org/wiki/Jacobi_eigenvalue_algorithm

[15] Online Source, *Divide-and-conquer eigenvalue algorithm.*,
http://en.wikipedia.org/wiki/Divide-and-conquer_eigenvalue_algorithm

[16] Arbenz, Dr. peter *Lecture Notes*, http://people.inf.ethz.ch/arbenz/ewp/Lnotes/chapter4.pdf

[17] Kumar, Dr. Sabheev *Krylov Subspace Methods for the Eigenvalue problem*,
https://cseweb.ucsd.edu/classes/fa04/cse252c/sakumar.pdf

[18] Online Source, *Arnoldi Iteration*, http://en.wikipedia.org/wiki/Arnoldi_iteration

[19] Online Source, *Lanczos Iteration*, http://en.wikipedia.org/wiki/Lanczos_algorithm

[20] Online Source, *Rayleigh Quotient*, http://en.wikipedia.org/wiki/Rayleigh_quotient

[21] L. Harris, I. Papanikolaou, J. Shi, D. Strott, Y. Tan and C. Zhong
Eigensystems for Large Matrices, AMSC460 - Spring 2015
https://goo.gl/EaldQv