CONTINUATION ALGORITHMS FOR THE SOLUTION
OF THE EIGENVALUE PROBLEM

by

BOBBY GREEN, B.S. in E.E.

A THESIS

IN

ELECTRICAL ENGINEERING

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for
the Degree of

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

Approved

Accepted

August, 1979
ACKNOWLEDGEMENT

I would like to thank Dr. Richard Saeks for his assistance and guidance in the origination and completion of this thesis.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>ii</td>
</tr>
<tr>
<td>ONE. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>TWO. CLASSICAL METHODS FOR EIGENVALUE COMPUTATIONS</td>
<td>4</td>
</tr>
<tr>
<td>THREE. SPARSE MATRIX METHODS FOR EIGENVALUE COMPUTATION</td>
<td>10</td>
</tr>
<tr>
<td>FOUR. LU CONTINUATION ALGORITHM</td>
<td>19</td>
</tr>
<tr>
<td>FIVE. QU CONTINUATIONS ALGORITHM</td>
<td>28</td>
</tr>
<tr>
<td>SIX. A HESSENBERG CONTINUATION ALGORITHM</td>
<td>33</td>
</tr>
<tr>
<td>SEVEN. A NUMERICAL EXAMPLE</td>
<td>37</td>
</tr>
<tr>
<td>EIGHT. CONCLUSION</td>
<td>39</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>40</td>
</tr>
</tbody>
</table>
CHAPTER ONE

INTRODUCTION

The notion of a continuation algorithm for solving the eigenvalue problem arises in the study of large interconnecting systems. The size of the system and the number of interconnections in general will produce a model which can be written as a large sparse system of linear equations. Moreover, in many problems, this system of equations is parameterized by some variable, frequency, a variable component parameter, etc.

A particular problem which arises in the analysis of large scale systems is the determination of their stability characteristics in one form or another. This usually requires computation of the eigenvalues of the system of equations with which the system is modeled, often as a function of the underlying variable parameter. In particular, this is the case for the root locus and Nyquist criteria [3,7,8].

Classically, the solution of such a parameterized family of eigenvalue problems is achieved by discretizing the variable parameter and repeatedly solving the resultant eigenvalue problem via standard sparse matrix methods. Repeated solution of the eigenvalue problem, even with the computational overhead prorated among the various solutions, proves to be time consuming and laborious. An alternative approach is to solve the classical eigenvalue only once for
some initial parameter value and to integrate a differential equation whose solution characterizes the eigenvalues as a function of the variable parameter. Such continuation algorithms have proven to be effective techniques for the inversion of parameterized families of sparse matrices, the solution of nonlinear equations, large change sensitivity analysis, the D.C. analysis of linear systems, etc. Indeed, a continuation algorithm for the solution of the eigenvalue problem [11] unfortunately requires that one work with both the eigenvectors of the given system of equations and those of the transposed system which typically are non-sparse even for sparse systems. As such, this algorithm is not applicable in the present context.

In this thesis, we will present a continuations algorithm for tracking the eigenvalues of a large sparse system of equations. This is achieved by constructing a family of similarity transformations which triangularize the given system as a function of the underlying parameter. Since both the resultant triangular matrix and the similarity transformations themselves retain the sparseness of the given system of equations, the resultant algorithm proves to be quite efficient when applied to our large scale system problems. Of course, the eigenvalues of the given family of systems are just the diagonal entries of the resultant triangular systems.
In the following chapter, we survey some of the methods which have classically been employed for the solution of the eigenvalue problem. This is followed in Chapter 3 by a survey of some of the standard methods for computing the eigenvalues of a single large sparse system of equations. Finally, Chapters 4, 5 and 6 are devoted to the derivation of three alternative continuation algorithms: an LU continuations algorithm, a QR continuations algorithm and a Hessenberg continuations algorithm. A numerical example is presented in Chapter 7, while Chapter 8 is devoted to a discussion of the work and suggestions for further work.
In the classical eigenvalue problem, we are faced with the problem of finding the eigenvalues for a system of linear equations which may be expressed as (1).

\[ Ax = Y \]  \hspace{2cm} (1)

in matrix form \([1]\). In order to find the eigenvalues of the system, we simply let \( Y = \lambda x \), and we solve

\[ Ax = \lambda x \]  \hspace{2cm} (2)

or equivalently,

\[ (A-\lambda I)x = 0 \]  \hspace{2cm} (3)

where \( x \neq 0 \). Here, \( I \) is the \( nxn \) identity matrix. Now (3) has a non zero solution \( x \) if and only if the matrix \( (A-\lambda I) \) is singular. Hence, we may solve for the eigenvalues without explicitly computing the eigenvectors by finding the determinant of \( (A-\lambda I) \), termed the characteristic polynomial for \( A \) and computing its zeros. If \( A \) is an \( nxn \) matrix, then the characteristic polynomial will be an \( n^{th} \) order polynomial whose \( n \) complex roots are the eigenvalues of \( A \). Of course, once the eigenvalues are known, the eigenvectors of \( A \) can be obtained from (3).
In practice, the computation of the characteristic polynomial for a matrix is numerically intractable as is the computation of its roots. As such, many techniques for the solution of the eigenvalue problem involve some sort of similarity transformation which reduces the given system of equations to an equivalent system whose eigenvalues are more readily computed [11]. In particular, if \( R \) is a nonsingular matrix such that \( RR^{-1} = I \), then one can perform a similarity transformation on \( A \) by letting

\[
A' = RAR^{-1}
\]  
(4)

In order to show that the similarity transformation does not change the eigenvalues of the given system, we observe that

\[
(A' - \lambda I) = (RAR^{-1} - \lambda I) = (RAR^{-1} - \lambda RIR^{-1})
\]

\[
(A' - \lambda I) = R(A - \lambda I)R^{-1}
\]  
(5)

hence,

\[
det(A' - \lambda I) = det\{ (R)(A - \lambda I)(R^{-1}) \}
\]

\[
= det(R) \ det(A - \lambda I) \ det(R^{-1})
\]

\[
= det(R) \ det(A - \lambda I) \frac{1}{det(R)}
\]

\[
= det(A - \lambda I)
\]  
(6)

Showing that \( A' \) and \( A \) have the same characteristic polynomials, thus the same eigenvalues.
Note that such a similarity transformation may change the eigenvectors of the given system of equations, but once the eigenvalues have been computed via the similarity transformation, it is usually a straightforward process to go back to the original system to compute its eigenvectors if needed.

From the point of view of eigenvalue computation, a most advantageous similarity transformation is one that will leave a diagonal $A'$ matrix. In this case, the eigenvalues are just the trace elements of $A'$. Unfortunately, the $R$ matrix to achieve such a similarity transformation has the eigenvectors of $A$ as its columns. As such, the technique is not applicable to the sparse matrix problem, though it may be used to compute the eigenvalues of a non-sparse matrix.

A second similarity transformation of interest is one in which $R$ is chosen in such a way that when we operate on $A$, we get a triangular matrix. As before, the eigenvalues of the original system of equations are given by the diagonal entries of the resultant triangular matrix $A'$. In this case, however, if the given system of equations is sparse, the transformation may be carried out with a sparse $R$ and will result in a sparse $A'$.

A third class of similarity transformation of interest transforms $A$ into an $A'$, which is in Hessenberg form, i.e.,
"almost triangular" form. Although the computation of the eigenvalues of a matrix in Hessenberg form is not as simple as computing the eigenvalues of a diagonal on a triangular matrix, we have nonetheless reduced the complexity of the problem a great deal.

A number of algorithms for computing the eigenvalues of a single non-sparse matrix are presented by Wilkinson [11], Boedwig [1], Twerson [10] and Householder [5].

One such procedure allows one to compute the characteristic polynomial of $A$ via a numerical process resembling Gaussian elimination [9]. Once computed, one of the standard algorithms is then used for computing the zeros of the resultant characteristic polynomials to solve for the eigenvalues of the given system.

A second method is the power method, $x^{(p)} = A^p V$, in which $V$ is an "almost arbitrary" starting vector and produces the dominant eigenvalues with their eigenvectors.

Indeed, for large values of $p$, $x^{(p)}$ approximates the eigenvector corresponding to the dominant eigenvalue

$$\lambda \approx x^{(p)T} A x^{(p)} / x^{(p)T} x^{(p)}$$

where "$T$" denotes the transpose vector. This formula for $\lambda$ is known as the Raleigh quotient and modifications of the process lead to the absolutely smallest and to certain next dominant eigenvectors.
The Jacoby method subjects a real symmetric matrix to a sequence of orthogonal similarity transformations, each of which is based on a rotation matrix.

\[ Q_R = \begin{pmatrix} \cos \phi_R & -\sin \phi_R \\ \sin \phi_R & \cos \phi_R \end{pmatrix}. \] (8)

After \( n \) such steps, \( A \) will be transformed into

\[ Q_n^{-1} Q_n^{-1} \cdots Q_1^{-1} A Q_1 Q_2 \cdots Q_n \] (9)

and with proper selection of the \( \phi_R \)'s, this approach yields a diagonal form where

\[ R = Q_n^{-1} Q_{n-1}^{-1} Q_{n-2}^{-1} \cdots Q_1^{-1} \] (10)

where \( R \) is the required similarity transformation.

The Givens method is a similarity transformation to reduce \( A \) to tridiagonal form, and it achieves this in a finite number of steps. It then generates the characteristic polynomial which then provides a Sturm sequence for finding the real roots. The eigenvectors then follow easily via the Jacoby method.

As an example, consider the application of the characteristic polynomial approach to the solution of the eigenvalue problem for the system of equations

\[
\begin{align*}
(2-\lambda)x_1 &- x_2 + 0 \cdot x_3 = 0 \\
x_1 &+ (2-\lambda) x_2 - x_3 = 0 \\
0 &x_1 + x_2 + (2-\lambda) x_3 = 0
\end{align*}
\] (11)
The first step is to take linear combinations of equations until only the $x_3$ column of coefficients involves $\lambda$. For example, if $E_1$, $E_2$ and $E_3$ denote the three equations, then $-E_2 + \lambda E_1$ is the equation

$$x_1 - 2x_2 + (1-2\lambda-\lambda^2)x_3 = 0 \quad (12)$$

call this $E_4$ and the combination $E_1 - 2E_3 + \lambda E_4$ becomes

$$4x_1 - 5x_2 + (2+\lambda+2\lambda^2-\lambda^3)x_3 = 0 \quad (13)$$

The last two equations, along with $E_3$, now involve $\lambda$ in only the $x_3$ coefficients. Now we triangularize the system by Gaussian elimination obtaining

$$(4-10\lambda+6\lambda^2-\lambda^3)x_3 = 0 \quad (14)$$

or equivalently,

$$(4-10\lambda+6\lambda^2-\lambda^3) = 0 \quad (15)$$

if $x_3 \neq 0$ this may be solved for the eigenvalues

$$\lambda = 2, 2+\sqrt{2}.$$
CHAPTER THREE
SPARSE MATRIX METHODS FOR
EIGENVALUE COMPUTATION

The key factor with which one must deal when working with sparse matrices is that the inverse of a sparse matrix is "almost always" non sparse. As such, if we are to employ a similarity transformation as an aid in the solution of the eigenvalue problem for a sparse matrix, $R$ must be chosen so as to lie in one of those narrow classes of sparse matrices which admit a sparse inverse, since both $R$ and $R^{-1}$ are required for the similarity transformation. In particular, it has been found that triangular and orthogonal similarity transformations are best suited for the solution of the eigenvalue problem for sparse matrices.

Since the given system of equations typically falls into neither of these classes, as a first step in the solution of the eigenvalue problem, the given matrix $A$ is factored into the product of triangular and/or orthogonal matrices.

Two factorization algorithms that will be of interest to us are the LU factorization and the QU factorization (sometimes termed the "QR factorization") \[11\].

The LU factorization is a method of factoring any matrix $A$ into the product of two other matrices, one a lower triangular matrix $L$ and another an upper triangular
matrix $U$ such that

$$A = LU$$

(1)

One method for factoring $A$ is called the Crout Tewerson [10]. Let the $(i,j)$ elements of $L$ and $U$ be denoted $l_{ij}$ and $u_{ij}$, respectively. Let us construct $U$ as a unite upper triangular matrix; that is, $u_{kk}=1, k=1,2,...n$. Now let us assume that the first $k-1$ rows and columns of $L$ and $U$ have already been determined. Then we have $l_{ip} = 0$ for $p>i$, $u_{pk} = 0$ for $p>k$ and $u_{kk} = 1$ and

$$a_{ik} = l_{ik} + \sum_{p=1}^{k-1} l_{ip}u_{pk}, \ i>k,$$

(2)

which gives

$$l_{ik} = a_{ik} - \sum_{p=1}^{k-1} l_{ip}u_{pk}, \ i>k.$$

(3)

Thus, the $k^{th}$ column of $L$ is now known. Now, from (1) and the fact that $l_{kp} = 0$ for $p>k$, we have

$$a_{kj} = l_{kk}u_{kj} + \sum_{p=1}^{k-1} l_{kp}u_{pj}, \ j>k$$

(4)

which yields

$$u_{kj} = (a_{kj} - \sum_{p=1}^{k-1} l_{kp}u_{pj})/l_{kk}, \ j>k$$

(5)

and the $k^{th}$ row of $U$ is now known. Thus, we have the first $(k-1)$ rows of $U$ and the first $(k-1)$ columns of $L$
and the $k$th rows of $U$ and the $k$th column of $L$ can be computed. The first row of $L$ is given by

$$ l_{i1} = a_{i1}, \quad i = 1, 2, \ldots, n, $$

(6)
due to the fact that the first column of $U$ is $e_1 = (1, 0, 0, \ldots, 0)^T$. The first row of $U$ is easy to find due to the fact that the first row of $L$ is $l_{11} e_1^T$, namely,

$$ u_{1j} = a_{1j} / l_{11}, \quad j > 1 $$

(7)

The QU factorization is similar to the LU factorization in that it is another factorization routine. However, rather than being lower and upper triangular, the QU factorization factors a matrix $A$ into the product of an orthogonal and a triangular matrix such that $Q$ is orthogonal and $U$ is upper triangular.

$$ A = QU $$

(8)

Note that for both the LU and QU factorizations, the factors of a sparse matrix are themselves sparse. The most obvious application of the above factorization is as an aid in matrix inversion.

Here, one expresses a sparse matrix $A$ as

$$ A = LU $$

(9)
\[ A = QU \quad (10) \]

and then writes \( A^{-1} \) as

\[ A^{-1} = U^{-1}L^{-1} \quad (11) \]

or

\[ A^{-1} = U^{-1}Q^{-1} \quad (12) \]

Since both triangular matrices and orthogonal matrices are readily inverted, \( A^{-1} \) is easily computed by this method. Moreover, since the inverse of sparse triangular matrices and sparse orthogonal matrices are sparse, the expressions of equations (11) and (12) for \( A^{-1} \) preserve the sparseness of \( A \) even though \( A^{-1} \) may not be sparse.

To apply the LU and QU factorizations to the eigenvalue problem for sparse matrices, we employ a sequence of factorizations to construct a similarity transformation which will triangularize the given system of equations in the limit.

The first approach uses a sequence of LU factorizations to construct an upper triangular similarity transformation. Here we let [1]

\[
\begin{align*}
A &= A_0 = L_0 U_0 \\
A_1 &= U_0 L_0 = L_1 U_1 \\
A_2 &= U_1 L_1 = L_2 U_2 \\
&\vdots
\end{align*}
\]
Now if \( A \) is invertable, then both \( L \) and \( U \) will be invertable. Hence, if we let

\[
L_0 = AU_0^{-1}
\]  

(14)

we obtain

\[
A_1 = U_0AU_0^{-1} = L_1U_1.
\]  

(15)

Now let

\[
L_1 = AU_1^{-1}
\]  

(16)

obtaining

\[
A_2 = U_1A_1U_1^{-1}
\]  

(17)

and

\[
A_2 = U_1U_0AU_0^{-1}U_1^{-1}
\]  

(18)

Now, if we continue the process and assume that

\[
\text{limit } \lim_{n \to \infty} U_{n-1} = \lim_{n \to \infty} U^n = U
\]  

(19)

converges. Then,

\[
\lim_{i \to \infty} U_i = I
\]  

(20)
Then we can write

\[ A_{n+1} = U_n U_{n-1} \ldots U_0 A U_0^{-1} U_1^{-1} \ldots U_n^{-1} \]  
\[ = U_n A (U_n)^{-1} = L_{n+1} U_{n+1} \]  

As such, \( A_{n+1} \) is a similarity transformation of \( A \) by a unit triangular matrix. Moreover,

\[ \lim_{n \to \infty} A_{n+1} = \lim_{n \to \infty} L_{n+s} U_{n+1} \]
\[ = [\lim_{n \to \infty} L_{n+1}] [\lim_{n \to \infty} U_{n+1}] \]
\[ = \lim_{n \to \infty} L_{n+1} \]  

is lower triangular, since it is the limit of lower triangular matrices. Thus, by application of a sequence of \( LU \) factorizations, we obtain a sequence of unit upper triangular similarity transformations which asymptotically transform the given system of equations to a lower triangular system of equations whose eigenvalues are readily computed. The resultant "LU algorithm" preserves the sparseness of \( A \) at every step and has been shown to be quite effective for the solution of the eigenvalue problem given a sparse \( A \).

The "QU algorithm" is similar to the LU algorithm, except that a QU factorization is used in lieu of the LU factorization. This in turn results in a sequence of
orthogonal similarity transformations which asymptotically transform the given system of equations into a triangular system. As before, the sparseness of $A$ is preserved at every step. Moreover, the "condition" $[1,11]$ of $A$ is also preserved at every step by virtue of the fact that the similarity transformations which are employed are orthogonal.

A third approach to the solution of the eigenvalue problem constructs a triangular similarity transformation which transforms a given sparse system of equations into a Hessenberg form [1]. We will illustrate a method for constructing the similarity transformation taken from Bodewig, and as we will see, the simplicity gained in constructing the transformation will be offset by the increased difficulty in determining the eigenvalues. The advantage of the resultant Hessenberg form is that it will yield eigenvalues more readily than the original $A$, using an easily computed similarity transformation.

We start with an arbitrary vector $v_0$ and form $Av_0$ [1]. Then we choose $c_{11}$ so that in

$$v_1 = Av_0 - c_{11}v_0$$

(23)

the first component vanishes. Then $c_{21}$ and $c_{22}$ are chosen so that in

$$v_2 = Av_1 - c_{12}v_0 - c_{22}v_1$$

(24)
the first two components vanish. Generally, \( c_{1m}, c_{2m}, \ldots \) \( c_{mm} \) have to be chosen so that in

\[
v_m = A v_{m-1} - c_{1m} v_0 - c_{2m} v_1 - \ldots - c_{mm} v_{m-1}
\]

(25)

the first \( m \) components vanish. Therefore, the \( n \) vectors \( v_i, i = 0,1,\ldots,n-1 \) gives us a triangular matrix

\[
V = (v_0, v_1, v_2, \ldots v_{n-1})
\]

(26)

Now

\[
A v_{(m-1)} = V(c_{1m}, c_{2m}, \ldots, c_{mm}, 1, \ldots, 0)^T,
\]

(27)

Thus,

\[
AV = VC
\]

(28)

when,

\[
C = \begin{bmatrix}
  c_{11} & c_{12} & \cdots & c_{1n} \\
  1 & c_{22} & c_{23} & \cdots & c_{2n} \\
  0 & 1 & c_{33} & \cdots & \cdot \\
  0 & 0 & \cdots & \cdot & \cdot \\
  0 & 0 & \cdots & \cdot & 1 & c_{nn}
\end{bmatrix}
\]

(29)

and
As such, if \( v_i^{i+1} \neq 0 \) for \( i = 1, 2, \ldots, n-1 \), \( V^{-1} \) exists. This gives us a triangular similarity transform \( V \), which transforms the given system of equations \( A \) into upper Hessenberg form via

\[
V^{-1}AV = C
\]
CHAPTER FOUR

LU CONTINUATION ALGORITHM

In the preceding chapters, we have described some of the standard methods for computing the eigenvalues of a matrix. In many applications however, one desires to compute the eigenvalues of a continuously parameterized family of matrices, \( A(r) \), as a function of \( r \). Although one can achieve this end by discretizing \( r \) and solving the classical eigenvalue problem repeatedly, we would prefer to formulate an algorithm which exploits the continuous parameterization of \( A(r) \) by \( r \). One class of algorithms which can achieve this end are the continuation algorithms wherein we formulate a differential equation

\[
\frac{dx}{dr} = F(x,r); \ x(r_0), \text{ given.} \tag{1}
\]

Where \( x(r) \) represents the eigenvalues of \( A(r) \) and/or a set of auxiliary variables from which the eigenvalues of \( A(r) \) can be computed (eigenvectors, a similarity transformation which triangularizes \( A(r) \), etc.). Once such a differential equation has been formulated one computes \( x(r_0) \) by classical means and integrates the differential equation to obtain \( x(r) \) for the remaining parameter values.

Such continuation algorithms have proved to be successful in a number of applications including the inversion of sparse matrices \([2,6,12]\). Moreover, a continuation al-
gorithm for the solution of the eigenvalue problem for a family of non-sparse matrices has been given by Faddeev and Faddeva [4]. In the present section, we present the first of three new continuation algorithms for the solution of the eigenvalue problem for a family of large sparse matrices. These are based on the LU algorithm, the QU algorithm and the Hessenberg algorithm described in the preceding chapters.

Rather than formulating a differential equation directly, these algorithms will be presented in terms of a finite difference notation which is more suitable to numerical implementation. Of course, a differential equation can be derived by taking a limiting case of the finite difference formula.

For the sake of notational brevity, we will denote $A(r_0)$ by $M$ and $A(r_0 + \Delta r)$ by $M + \varepsilon^0$. We then assume that we have a similarity transformation on $M$ by means of some $R$ and $R^{-1}$ which allows us to compute the eigenvalues of $M$ and we proceed to formulate a similarity transformation for computing the eigenvalues of $M + \varepsilon^0$ via a finite difference analysis. Of course, by the repeated application of such an analysis, one can compute the eigenvalues of $A(r_0 + \Delta r)$, $A(r_0 + 2\Delta r)$, ..., $A(r_0 + n\Delta r)$ iteratively.

Our first such algorithm is a continuations version of the LU algorithm. Here we assume that a unit upper
triangular matrix $U^0$ is given such that

$$U^0 M U^0^{-1} = L^0. \quad (2)$$

where $L^0$ is lower triangular.

Note that such a $U^0$ may be computed by the classical $LU$ algorithm. Now we desire to find a new unite upper triangular matrix

$$U^1 = U^0 + \Delta^0 \quad (3)$$

which transforms $A(r_0 + \Delta r) = M + \epsilon^0$ into a new lower triangular matrix

$$L^1 = U^1 [M + \epsilon^0] U^1^{-1}. \quad (4)$$

Now substitute (3) into (4) and obtain

$$L^1 = [U^0 + \Delta^0] [M + \epsilon^0] [U^0 + \Delta^0]^{-1}, \quad (5)$$

For small $\delta$

$$[1 + \delta]^{-1} = [1 - \delta + \delta^2 - \ldots] \approx 1 - \delta. \quad (6)$$

As such

$$[U^0 + \Delta^0]^{-1} = [U^0 [I + U^0^{-1} \Delta^0]]^{-1} \quad (7)$$

and by letting $U^0^{-1} \Delta^0 = \delta$ we have

$$[U^0 + \Delta^0]^{-1} = [1 - U^0^{-1} \Delta^0 + (U^0^{-1} \Delta^0)^2 - (U^0^{-1} \Delta^0)^3 + \ldots] U^0^{-1}. \quad (8)$$
Neglecting second and higher order terms

\[[U^0 + \Delta^0]^{-1} - [1 - U^0 - \Delta^0]U^0 - 1 \]

= \[[U^0 - 1 - U^0 - \Delta^0 U^0 - 1]\].

Substituting (10) into (5) we obtain

\[L^1 = (U^0 + \Delta^0)(M + \epsilon^0)(U^0 - 1 - U^0 - \Delta^0 U^0 - 1).\]

Expanding

\[L^{-1} = (U^0M + U^0\epsilon^0 + \Delta^0M + \Delta^0\epsilon^0)(U^0 - 1 - U^0 - \Delta^0 U^0 - 1)\]

\[= (U^0MU^0 - 1 + U^0\epsilon U^0 - 1 + \Delta^0MU^0 - 1 + \Delta^0\epsilon U^0 - 1)\]

\[- U^0MU^0 - 1 \Delta^0 U^0 - 1 - (U^0\epsilon + \Delta^0M + \Delta^0\epsilon^0)(U^0 - 1 \Delta^0 U^0 - 1)\]

(12)

and neglecting all second and higher order terms we have

\[L^1 = U^0MU^0 - 1 - U^0MU^0 - 1 \Delta^0 U^0 - 1\]

\[+ U^0\epsilon U^0 + \Delta^0MU^0 - 1\]

(13)

Substituting the product \(UU^{-1}\) into the last term of (13) we obtain

\[L^1 = U^0MU^0 - 1 - U^0MU^0 - 1 \Delta^0 U^0 - 1 + U^0\epsilon U^0 + \Delta^0U^0 - 1 U^0MU^0 - 1\]

(14)

and recalling (2) we have

\[L^1 = L^0 - L^0 \Delta^0 U^0 - 1 + U^0\epsilon U^0 - 1 + \Delta^0U^0 - 1 L^0\]

(15)

letting \(\Delta^0U^0 - 1 = Z^0\) we have
\[ L^1 = L^0 - L^0 Z^0 + U^0 - 1 + Z^0 L^0. \]  

(16)

Defining \( u[A] \) to be the upper triangular portion of \( A \) without the diagonal we may write equation (16) obtaining

\[ u[L^1] = u[L^0 + Z^0 L^0 + U^0 e^0 U^0 - 1 - L^0 Z^0]. \]  

(17)

Now, since the \( u[\cdot] \) operator is a linear operator, we can write

\[ u[L^1] = u[L^0] + u[U^0 e^0 U^0 - 1] + u[Z^0 L^0 - L^0 Z^0] \]  

(18)

and since

\[ u[L^1] = u[L^0] = 0 \]  

(19)

we arrive at

\[ 0 = u[Z^0 L^0 - Z^0 L^0] + u[U^0 e^0 U^0 - 1] \]  

(20)

Rearranging and changing signs so that

\[ u[U^0 e^0 U^0 + 1] = u[L^0 Z^0 - Z^0 L^0] \]  

(21)

and introducing the commutator operator \([L, Z]\) we get

\[ u[U^0 e^0 U^0 - 1] = u[[L, Z]]. \]  

(22)

Since both \( U^0 \) and \( U^1 \) are required to be unite upper triangular matrices, it follows that \( \Delta^0 = U^1 - U^0 \) is strictly upper triangular. That \( \Delta^0 \) is an upper triangular with zeros on the diagonal and thus represents \((n(n-1))/2\) non trivial unknowns in equation (22). Similarly, the \( u[\cdot] \) operator zeros out the lower triangular entries of equation (22), thereby leaving \((n(n-1))/2\) nontrivial eqns. As such,
equation (22) represents a set of \( \frac{n(n-1)}{2} \) linear equations and \( \frac{n(n-1)}{2} \) unknowns which can be solved for \( Z^0 \) and hence \( A^0 = Z^0 U^0 \). Unfortunately we cannot solve for \( Z \) in this form. It is however, a simple matter to solve for \( Z \) term by term by expanding the commutator.

To begin with an example, let

\[
U^0 e^0 U^{0-1} = Y^0
\]  

(23)

and then expand a 4\times4 set of equations of the form

\[
u[Y^0] = \nu[[L^0, Z^0]]
\]

\[
\begin{bmatrix}
\lambda_{11} & 0 & 0 & 0 \\
\lambda_{21} & \lambda_{22} & 0 & 0 \\
\lambda_{31} & \lambda_{32} & \lambda_{33} & 0 \\
\lambda_{41} & \lambda_{42} & \lambda_{43} & \lambda_{44}
\end{bmatrix}
\begin{bmatrix}
0 & Z_{12} & Z_{13} & Z_{14} \\
0 & 0 & Z_{23} & Z_{24} \\
0 & 0 & 0 & Z_{34} \\
0 & 0 & 0 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & Z_{12} & Z_{13} & Z_{14} \\
0 & 0 & Z_{23} & Z_{24} \\
0 & 0 & 0 & Z_{34} \\
0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\lambda_{11} & 0 & 0 & 0 \\
\lambda_{21} & \lambda_{22} & 0 & 0 \\
\lambda_{31} & \lambda_{32} & \lambda_{33} & 0 \\
\lambda_{41} & \lambda_{42} & \lambda_{43} & \lambda_{44}
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & Y_{12} & Y_{13} & Y_{14} \\
0 & 0 & Y_{23} & Y_{24} \\
0 & 0 & 0 & Y_{34} \\
0 & 0 & 0 & 0
\end{bmatrix}
\]

(24)
Now solving for the $z_{ij}$'s term by term and letting $\lambda_{ij} =
\lambda_{ii} - \lambda_{jj}$, where the $\lambda_{ii}$'s are the eigenvalues of $M$
results in the following set of equations

\[
\begin{pmatrix}
\lambda_{34} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \lambda_{23} & -\lambda_{43} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \lambda_{24} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \lambda_{12} & -\lambda_{32} & -\lambda_{42} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \lambda_{13} & -\lambda_{43} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \lambda_{14} & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
z_{34} \\
z_{23} \\
z_{24} \\
z_{12} \\
z_{13} \\
z_{14}
\end{pmatrix}
= 
\begin{pmatrix}
y_{34} \\
y_{23} \\
y_{24} \\
y_{12} \\
y_{13} \\
y_{14}
\end{pmatrix}
\] (25)

We can see that (25) is a lower triangular set of equations that can easily be inverted to from eqn. (26) where
we are inverting a sparse triangular matrix to solve for
the $z_{ij}$ entries in $Z$. Indeed, this matrix can be inverted
analytically so that we can write the $z_{ij}$'s as a very con-
vlenent and unique sum.

\[
\begin{align*}
    z_{14} & = [y_{14}] / \lambda_{14} \\
    z_{13} & = [y_{13} + z_{14} \lambda_{43}] / \lambda_{13} \\
    z_{12} & = [y_{12} + z_{14} \lambda_{42} + z_{13} \lambda_{32}] / \lambda_{12} \\
    z_{24} & = [y_{24} - \lambda_{21} z_{14}] / \lambda_{24} \\
    z_{23} & = [y_{23} - \lambda_{21} z_{14} + z_{24} \lambda_{43}] / \lambda_{23} \\
    z_{34} & = [y_{34} - \lambda_{31} z_{14} - \lambda_{32} z_{24}] / \lambda_{34}
\end{align*}
\] (26)
For an $n \times n$ matrix $M$ a similar derivation yields

$$z_{ij} = \frac{\sum_{n=1}^{i-1} \lambda_i z_{n,j}^i + \sum_{k=j+1}^{n} z_{k,j}^i \Delta^k}{\lambda_{ij}'}$$

(27)

which are well defined as long as $\lambda_{ij} = \lambda_{ii} - \lambda_{jj}$ does not equal zero, i.e., the matrix $M$ has distinct eigenvalues.

Note that $z_{ij}$ may be computed sequentially by first letting $i = 1$, and letting $j$ run from $n$, $n-1$, ..., $2$, we then let $i = 2$, and let $j$ run from $n$, $n-1$, ..., $3$. Repeating the process we increment $i$ and let $j$ run from $n$ through $i+1$ until we have computed all the $z_{ij}$'s, $i < j$.

Now that we have solved for $Z^0$, let us rewrite equation (12) in such a manner that will permit us to formulate an iterative algorithm for computing $A(r)$ as a function of $r$.

We begin with

$$L^1 = [U^0 + \Delta^0] [M + \Delta^0] [U^0^{-1} - U^0^{-1} \Delta^0 U^0^{-1}]$$

(12)

and factor out $U^0$ and $U^0^{-1}$ in the appropriate manner to obtain

$$L^1 = [(1+\Delta^0)U^0^{-1}]U^0[M+\Delta^0]U^0^{-1}[1-\Delta^0 U^0^{-1}]$$

(28)

Substituting $Z^0 = \Delta^0 U^0^{-1}$

$$L^1 = [(1+Z^0)U^0[M+\Delta^0]U^0^{-1}[1-Z^0]$$

(29)

and multiplying through by $U^0$ and $U^0^{-1}$ we obtain
\[ L^1 = [1+Z^0][U^0MU^0^{-1} + U^0\varepsilon^0U^0^{-1}] [1-Z^0] \] (30)

and

\[ L^1 = [1-Z^0][L^0+Y^0][1-Z^0]. \] (31)

In order to iterate this routine, all that is necessary is an initial starting point, \( M_0 \). Then we remember that the updated similarity transformation is \( U^{i+1} = [1+Z^i]U^i \) and that for each iteration \( L^i \) is replaced by \( L^{i+1} \) and \( Y^i \) is replaced by \( Y^{i+1} = U^{i+1}\varepsilon_{i+1}U^{i+1-1} \) where

\[ \varepsilon_i = A(r_0+(i+1)\Delta r) - A(r_0+i\Delta r) \] (32)

i.e. we have an updated similarity transform to operate on the present \( \varepsilon \).
CHAPTER FIVE
QU CONTINUATIONS ALGORITHM

In this chapter, we formulate a continuations algorithm for the solution of the eigenvalue problem for a continuously parameterized family of large sparse matrices which is based on the classical QU algorithm. Here we assume that \( X^0 \) is an orthogonal matrix which triangularizes our given initial matrix \( M \)

\[
X^0 M X^0 \mathsf{T} = L^0. \tag{1}
\]

We then assume that \( M \) is perturbed by a matrix \( \varepsilon^0 \) and we desire to find a new orthogonal matrix \( X^1 \), which triangularizes \( M + \varepsilon^0 \):

\[
X^1 [M + \varepsilon^0] X^1 \mathsf{T} = L^1 \tag{2}
\]

letting \( \Delta^0 = X^1 - X^0 \), we may rewrite equation (2) as

\[
[X^0 + \Delta^0] [M + \varepsilon^0] [X^0 + \varepsilon^0] \mathsf{T} = L^1

[X^0 + \Delta^0] [M + \varepsilon^0] [X^0 \mathsf{T} + \Delta^0] = L^1 \tag{3}
\]

Expanding (3), one obtains

\[
L^1 = [X^0 M + X^0 \varepsilon^0 + \Delta^0 M + \Delta^0 \varepsilon^0] [X^0 \mathsf{T} + \Delta^0 \mathsf{T}]

= [X^0 M X^0 \mathsf{T} + X^0 \varepsilon^0 X^0 \mathsf{T} + \Delta^0 M X^0 \mathsf{T} + \Delta^0 \varepsilon^0 X^0 \mathsf{T} + X^0 M \Delta^0 \mathsf{T}]

+ [X^0 \varepsilon^0 + \Delta^0 M + \Delta^0 \varepsilon^0] \Delta^0 \mathsf{T} \tag{4}
\]
Now assume that the $\epsilon$ is small, so that we can neglect second and higher order terms, in which case (4) reduces to

$$L^1 = [X^O MX^O + X^O \epsilon^O X^O + A^O MX^O + X^O M A^O]$$

(5)

Now we write

$$\Delta^O MX^O T = \Delta^O X^O T X^O MX^O T = \Delta^O X^O T L^O$$

(6)

and

$$X^O M A^O T = X^O MX^O T X^O A^O T = L^O X^O A^O T$$

(7)

obtaining

$$L^1 = [L^O + X^O \epsilon^O T + \Delta^O X^O T L^O + L^O \Delta^O X^O T]$$

(8)

Now let us determine the properties of $\Delta^O X^O T$ and $X^O \Delta^O T$ which are required to guarantee that $X^1$ will be orthogonal, given that $X^O$ is orthogonal. For $X^1$ to be orthogonal, we must have

$$X^1 X^1^T = [X^O + \Delta^O] [X^O + \Delta^O]^T = I$$

(9)

or equivalently,

$$[X^O + \Delta^O] [X^O^T + \Delta^O^T] = I.$$ 

(10)

Expanding, we obtain

$$I = X^O X^O^T + X^O \Delta^O T + \Delta^O X^O^T + \Delta^O \Delta^O T$$

(11)
and
\[ 0 = I - X^O X^O^T = X^O\Delta^O + \Delta^O X^O^T + \Delta^O \Delta^O^T \]  
(12)

Now, once again neglecting second and higher order terms, we have
\[ X^O\Delta^O^T = -\Delta^O X^O^T = -[X^O\Delta^O^T]^T \]  
(13)
defining \( Z \) by
\[ X^O\Delta^O^T = Z^O^T, \Delta^O X^O^T = Z^O \]  
(14)
we then have
\[ Z^O = -Z^O^T \]  
(15)
That is, \( Z^O \) is a skew symmetric matrix. Thus, by solving for \( Z_{ij} \), \( i<j \), we can reconstruct \( Z^O \) and \( \Delta^O = Z^O X^O \).

Now rewriting (8) and taking the upper triangular portion of the matrix without the diagonal, we obtain
\[ 0 = u[L^1] = u[L^O] + u[X^O\varepsilon X^O^T] + u[Z^O L^O - L^O Z^O]. \]  
(16)
Letting \( X^O\varepsilon X^O^T = Y^O \) yields
(17)
When we expand the \( u[[L^O, Z^O]] \) commutator term by term, we find that only the \( Z^O_{ij} \) elements for \( i<j \) appear. Indeed, a little analysis will reveal that we may solve for \( Z^O_{ij}, i<j \) by using exactly the same equations en-
countered in the LU continuations algorithm, i.e.,

\[ z_{ij}^o = (y_{ij}^o + \sum_{k=j+1}^{n} z_{i,k}^o k_{i,j} - \sum_{n=1}^{i-1} l_{i,n}^o z_{i,n}^o,k_{i,j})/\lambda_{ij}^o \]  

(18)

where \( \lambda_{ij}^o = l_{ii}^o - l_{jj}^o \) is the difference of the eigenvalues of \( \lambda_i^o \) and \( \lambda_j^o \) for our given matrix \( M \). As before, the equations can be solved sequentially by decreasing \( j \) and increasing \( i \) as long as the eigenvalues of the given matrix are distinct.

To formulate an iterative technique, one substitutes \( Z^o = \Delta^o X^o \) into (3);

\[ L^1 = [X^o+\Delta^o][M+\varepsilon^o][X^o_T+\Delta^o_T] \]

\[ = [1+\Delta^o X^o_T]X^o[M+\varepsilon]X^o_T[I+X^o \Delta^o_T] \]

obtaining

\[ L^1 = [1+Z^o]X^o[M+\varepsilon]X^o_T[I-Z^o] \]

(19)

\[ = [1+Z^o][L^o+Y^o][1-Z^o] \]

(20)

where \( Y = X^o \varepsilon X^o_T \).

Now, upon letting \( \varepsilon \) be

\[ \varepsilon^i = A(r^o-(i+1)\Delta r) - A(r^o+i\Delta r), \]

(21)

one may construct an iterative algorithm for computing the eigenvalues of \( A(r) \) by letting
\[ L^{i+1} = [1+Z^i] \{ L^i + Y^i \} [1-Z^i] \] (22)

\[ X^{i+1} + [1+Z^i]X^i \] (23)

where

\[ Y^i = x^i \xi^i x^i \] (24)

and \( Z^{i+1} \) is computed via equation (18) with appropriately updated indices.
In this chapter, we will formulate a continuations version of the Hessenberg algorithm discussed in Chapter Three. As before, we let $M = A(r_o)$ and $\epsilon^o = A(r_o + \Delta r) - A(r_o)$. We also assume that the algorithm of Chapter Three has been employed to determine an upper triangular matrix $V^o$ such that

$$V^o M V^o^{-1} = C^o$$  \hspace{1cm} (1)

where $C^o$ is in lower Hessenberg form with ones on the super diagonal, i.e., it is "almost lower triangular." It is now desired to determine a new upper triangular matrix $V^l$ such that

$$V^l [M + \epsilon^o] V^l^{-1} = C^l$$  \hspace{1cm} (2)

where $C^l$ is also in lower Hessenberg form with one's on the super diagonal.

Now, expressing $V^l$ in the form

$$V^l = V^o + \Delta^o$$  \hspace{1cm} (3)

and substituting into (2) we obtain

$$C^l = [V^o + \Delta^o][M + \epsilon^o][V^o + \Delta^o]^{-1}.$$  \hspace{1cm} (4)
Following the derivation of Chapter Four for small \( \Delta \) we may write

\[
[V^0 + \Delta^0]^{-1} \simeq [V^0]^{-1} - [V^0]^{-1} \Delta^0 [V^0]^{-1}
\]  

(5)
yielding

\[
C^1 = [V^0 + \Delta^0] [M + \varepsilon^0] [V^0]^{-1} [\Delta^0 [V^0]^{-1}]
\]

\[
= V^0 M^0 + V^0 \varepsilon V^0 - \Delta^0 M^0 - \Delta^0 \varepsilon V^0 - V^0 M^0 - \Delta^0 V^0
\]

(6)

Neglecting all second and higher order terms we obtain

\[
C^1 = V^0 M^0 - \varepsilon V^0 + \Delta^0 M^0 - V^0 M^0 - \Delta^0 V^0
\]

(7)

Now let us take the upper triangular portion of \( C^1 \) without the diagonal obtaining

\[
u[C^1] = \nu[V^0 M^0 - \varepsilon V^0] + \nu[\varepsilon V^0]
\]

(8)

Replacing \( V^0 M^0 \) with \( C^0 \) and noting that

\[
u[C^1] = \nu[C^0]
\]

(9)

we have

\[- \nu[V^0 \varepsilon V^0] = \nu[\Delta^0 M^0 - V^0 M^0 - \Delta^0 V^0]
\]

(10)

or equivalently,
Letting $Z = \Delta^O V^O$, we have

$$u[V^O \epsilon O V^O] = u[V^O \Delta^O V^O \epsilon O V^O] = u[\Delta^O V^O \epsilon O V^O] = u[V^O \Delta^O V^O \epsilon O V^O].$$  \hfill (11)

Letting $Y^O = V^O \epsilon O V^O$ and expanding $u[(C, Z)]$, we solve for $Z$ term by term. This results in a set of simultaneous equations

$$
\begin{bmatrix}
\lambda_{34} & 0 & C_{32} & 0 & 0 & C_{31} \\
0 & \lambda_{23} & -C_{43} & 0 & C_{12} & 0 \\
1 & -1 & \lambda_{24} & 0 & 0 & C_{12} \\
0 & 0 & 0 & \lambda_{12} & -C_{32} & -C_{42} \\
0 & 1 & 0 & -1 & \lambda_{13} & -C_{43} \\
0 & 0 & 1 & 0 & -1 & \lambda_{14}
\end{bmatrix}
\begin{bmatrix}
Z_{34} \\
Z_{23} \\
Z_{24} \\
Z_{12} \\
Z_{13} \\
Z_{14}
\end{bmatrix}
= 
\begin{bmatrix}
Y_{34} \\
Y_{23} \\
Y_{24} \\
Y_{12} \\
Y_{13} \\
Y_{14}
\end{bmatrix}
\hfill (13)

where as before $\lambda_{ij} = C_{ii} - C_{jj}$.

Although the matrix of equation (13) is not triangular as in the LU or the QU algorithms, it is highly sparse. Moreover, its inversion can be implemented via a two step process where one inverts a single lower dimensional non-triangular matrix and a higher dimensional triangular matrix. This inversion then yields the desired $Z^O$ from which we get $\Delta^O = Z^O V^O$. 
As with the previous algorithms, the above derivation can be incorporated into an iterative algorithm for computing the eigenvalues of a continuously parameterized family of sparse matrices $A(r)$. Here we let

$$
\varepsilon^i = A(r_o + (i+1)\Delta r) - A(r_o + i\Delta r) \quad (14)
$$

$$
\gamma^i = \gamma^i \varepsilon^i \gamma^i^{-1} \quad (15)
$$

and

$$
C^{i+1} = [1 + \gamma^i] [C^i + \gamma^i] [-\gamma^i] \quad (16)
$$

Where $\gamma^i$ is computed via equation (13) with appropriately updated indices.
CHAPTER SEVEN

A NUMERICAL EXAMPLE

To illustrate the numerical accuracy of the continuation algorithms presented, the LU algorithm was employed to compute the eigenvalues of the matrix

\[
A(r) = \begin{bmatrix}
1155 & 726 & 2244 & -44 & 176 & 25454 \\
0 & -5 & -18 & 0 & 0 & 0 \\
0 & 3 & 10 & 0 & 0 & 0 \\
26 & 246 & 786 & -1 & 46 & (582+22r) \\
-14 & 37 & 119 & 0 & 9 & 0 \\
-52 & -33 & -102 & 2 & -8 & -1146
\end{bmatrix}
\]

where \( r \) is allowed to vary from 0 to 1. The algorithm was successively run using 1 step, 10 steps, 100 steps and 1000 steps in the interval of integration. The result of these computations are tabulated below. In general, the numerical error resulting from these computations seems to decrease linearly with step size.
\[ M_{46} = 582 + 22r \]

<table>
<thead>
<tr>
<th>( \lambda_{11} )</th>
<th>( \lambda_{22} )</th>
<th>( \lambda_{33} )</th>
<th>( \lambda_{44} )</th>
<th>( \lambda_{55} )</th>
<th>( \lambda_{66} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda(0) )</td>
<td>( \lambda(1) )</td>
<td>( \lambda(1) )</td>
<td>( \lambda(1) )</td>
<td>( \lambda(1) )</td>
<td>( \lambda(1) )</td>
</tr>
<tr>
<td>11.0</td>
<td>11.0</td>
<td>11.0</td>
<td>11.0</td>
<td>11.0</td>
<td>11.0</td>
</tr>
<tr>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>4.0</td>
<td>4.0</td>
<td>4.0</td>
<td>4.0</td>
<td>4.0</td>
<td>4.0</td>
</tr>
<tr>
<td>3.0</td>
<td>6.5</td>
<td>7.287</td>
<td>6.5787</td>
<td>6.50787</td>
<td>6.500787</td>
</tr>
<tr>
<td>9.0</td>
<td>9.0</td>
<td>9.0</td>
<td>9.0</td>
<td>9.0</td>
<td>9.0</td>
</tr>
</tbody>
</table>
CHAPTER EIGHT

CONCLUSION

In the preceding chapters we have presented three continuation algorithms for the solution of the eigenvalue problem for a continuously parameterized family of large sparse matrices. Unlike the algorithm of Faddeev and Faddeva [4] in which the eigenvectors of the matrices involved serve as auxiliary variables, in the present algorithm the auxiliary variables take the form of the sparse matrix $Z$. As such, these algorithms exploit the sparse nature of the given family matrices. Moreover, the order of the resultant differential equation is of a lesser order than that of Faddeev and Faddeva by a factor of four.
REFERENCES


