Numerical Methods for Two Parameter Eigenvalue Problems

Philip A. Browne

April 25, 2008

Abstract

This thesis is concerned with numerical solutions of two parameter eigenvalue problems. We firstly show that the matrix form of two parameter eigenvalue problems can be decoupled using the Kronecker product at the expense of an increase in dimensionality. We then go on to consider Newton's method and observe that we can obtain quadratic convergence to eigenvectors and eigenvalues for close enough starting values. Subsequently, we derive sufficient conditions for Newton's method to be applied. In order to test different methods we construct model examples of two parameter eigenvalue problems and also consider a real world three point boundary problem. Finally we look at tensor Rayleigh quotient iteration and apply it to model examples. The experiments are conducted in Matlab.

Contents

Chapter 1

Introduction

A two parameter eigenvalue problem in matrix form is

$$A_1\mathbf{x} = \lambda B_1\mathbf{x} + \mu C_1\mathbf{x} \tag{1.1}$$
$$A_2\mathbf{y} = \lambda B_2\mathbf{y} + \mu C_2\mathbf{y} \tag{1.2}$$

where λ, $\mu \in \mathbb{C}$, $\mathbf{x}$ & $\mathbf{y}$ are complex vectors and A_i, B_i and C_i are given complex matrices for $i = 1, 2$.

The need for a numeric solver for the two parameter eigenvalue problem arises when we look at a number of different physical situations, particularly where the separation of variables technique is used to solve a discretised boundary value problem.

A typical example of where a two parameter eigenvalue problem occurs is in the three point boundary problem. This could be a problem of the form

$$-(p(x)y'(x))' + q(x)y(x) = (\lambda r(x) + \mu s(x))y(x)$$

subject to the three boundary conditions $y(a) = y(b) = y(c) = 0$, where $a < b < c$ and x lies in the range $a < x < c$ We shall consider an example of this form in §4.5. This can then be treated as a two parameter eigenvalue problem when we consider the two separate cases

$$-(p(x)y_1'(x))' + q(x)y_1(x) = (\lambda r(x) + \mu s(x))y(x) \tag{1.3}$$

with boundary values $y_1(a) = y_1(b) = 0$ for x in the range $a < x < b$ and

$$-(p(x)y_2'(x))' + q(x)y_2(x) = (\lambda r(x) + \mu s(x))y(x) \tag{1.4}$$

with boundary values $y_2(b) = y_2(c) = 0$ for x in the range $b < x < c$.

Many of these problems are examples of two parameter Sturm-Liouville problems in the form

$$-(p_i(x_i)y_i'(x_i))' + q_i(x_i)y_i(x_i) = (\lambda a_{i1}(x_i) + \mu a_{i2}(x_i))y_i(x_i)$$

with given boundary conditions. This form occurs when the separation constants cannot be decoupled. These equations can be discretised to give a two parameter eigenvalue problem in the matrix form (1.1) & (1.2).

Another example where two parameter eigenvalue problems arise is from the separation of variables technique. If we consider the Helmholtz equation $\Delta u + k^2 u = 0$ in $\mathbb{R}^2$ that arises in the modelling of the vibration of a fixed membrane and perform separation of variables on an elliptic domain we obtain the Mathieu and modified Mathieu equations

$$y_1''(x_1) + (2\lambda \cosh 2x_1 - \mu) y_1(x_1) = 0 \tag{1.5}$$
$$y_2''(x_2) + (2\lambda \cos 2x_2 - \mu) y_2(x_2) = 0 \tag{1.6}$$

that have to be solved simultaneously and hence form a two parameter eigenvalue problem. Here λ is the eigenvalue corresponding to the physical value k^2, whereas μ is a somewhat artificial eigenvalue as it arises as a separation parameter.

Two parameter eigenvalue problems are also found in matrix form in varying circumstances. For instance, when estimating electrical properties of a material from measurements or interdigital dielectrometry sensors, the properties of a material that has two layers will be the eigenvalues from an appropriate two parameter eigenvalue problem[1].

Two parameter eigenvalue problems also crop up during dynamic model updating. Consider a spring-mass model with known mass matrix such that the stiffness parameters of two springs have to be updated based on external measurements of their natural frequencies. The updated parameters are then the eigenvalues to be found from a two parameter eigenvalue problem[2].

It can also be shown that the eigenvalues of a certain two parameter eigenvalue problem can give the optimum value of the over-relaxation parameter ω in the Young-Frankel scheme for a separable elliptic PDE in two independent variables[1].

This thesis will look at different techniques for solving two parameter eigenvalue problems. In Chapter 2 we consider using a special form of matrix multiplication known as the Kronecker product to solve two parameter eigenvalue problems and discuss the implications of using this method. We shall derive model problems in order that we can test various different techniques.

In Chapter 3 we consider Newton's method applied to a two parameter eigenvalue problem and in particular we shall show that it is a well-defined method to use. Following that, in Chapter 4 we will break down Newton's method using the specific structure of a two parameter eigenvalue problem and in doing so we shall increase the efficiency of this method.

Finally, in Chapter 5 we shall consider tensor Rayleigh quotient iteration and show how it is an amalgamation of both the Kronecker product and Newton's method. We shall provide numerical results for all these techniques as well as giving the Matlab code that implements them.

Chapter 2

Theory of Two Parameter Eigenvalue Problems

In this chapter we shall show how all solutions to the classical form of the two parameter eigenvalue problem are also solutions to a coupled pair of equations, where the eigenvalues λ and μ are decoupled, using the Kronecker Product. We shall also derive a model problem which will be used to test different numerical techniques. The details of which are outlined below.

The traditional problem is as follows:

$$\begin{aligned} A_1\mathbf{x} &= \lambda B_1\mathbf{x} + \mu C_1\mathbf{x} && (2.1)\\ A_2\mathbf{y} &= \lambda B_2\mathbf{y} + \mu C_2\mathbf{y} && (2.2) \end{aligned}$$

where $\lambda,\ \mu \in \mathbb{C}$ and $\mathbf{x} \in \mathbb{C}^n\backslash\{\mathbf{0}\}, A_1, B_1$ and $C_1 \in \mathbb{C}^{n\times n}$, $\mathbf{y} \in \mathbb{C}^m\backslash\{\mathbf{0}\}$ and A_2, B_2 and $C_2 \in \mathbb{C}^{m\times m}$.
We shall show in §2.2 that these equations imply

$$\begin{aligned} \Delta_1\mathbf{z} &= \lambda\Delta_0\mathbf{z} && (2.3)\\ \Delta_2\mathbf{z} &= \mu\Delta_0\mathbf{z} && (2.4) \end{aligned}$$

where Δ_0, Δ_1 and Δ_2 are $(nm)\times(nm)$ dimensional matrices defined as

$$\begin{aligned} \Delta_0 &= B_1\otimes C_2 - C_1\otimes B_2 && (2.5)\\ \Delta_1 &= A_1\otimes C_2 - C_1\otimes A_2 && (2.6)\\ \Delta_2 &= B_1\otimes A_2 - A_1\otimes B_2 && (2.7) \end{aligned}$$

and

$$\mathbf{z} = \mathbf{x}\otimes\mathbf{y} \qquad (2.8)$$

with $\otimes$ denoting the Kronecker (or Tensor) product of two matrices discussed in §2.1.

2.1 The Kronecker product

Definition 2.1.1. *The Kronecker Product* $(\cdot \otimes \cdot) : \mathbb{C}^{m\times n} \times \mathbb{C}^{p\times q} \to \mathbb{C}^{mp\times nq}$ *is defined by*

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \cdots & a_{nn}B \end{bmatrix}$$

where we use the standard notation $(A)_{ij} = a_{ij}$.

The Kronecker product is a special case of the tensor product, and as such it inherits the properties of bilinearity and associativity, ie

$$(kA) \otimes B = A \otimes (kB) = k(A \otimes B)$$
$$A \otimes (B + C) = A \otimes B + A \otimes C$$
$$(A + B) \otimes C = A \otimes C + B \otimes C$$

We now establish a famous property of the Kronecker product, from [3].

Lemma 2.1.1 (Mixed Product Property)**.** *Let* $A \in \mathbb{C}^{m\times n}$, $B \in \mathbb{C}^{p\times q}$, $C \in \mathbb{C}^{n\times k}$, $D \in \mathbb{C}^{q\times r}$. *Then*

$$(A \otimes B)(C \otimes D) = (AC \otimes BD)$$

Proof. Let $A = [a_{ih}]$ and $C = [c_{hj}]$. Partitioning according to the sizes of B and D, $A \otimes B = [a_{ih}B]$ and $C \otimes D = [c_{hj}D]$. The i,j block of $(A \otimes B)(C \otimes D)$ is

$$\sum_{h=1}^{n} a_{ih}Bc_{hj}D = \left[\sum_{h=1}^{n} a_{ih}c_{hj}\right] BD$$

But this is just the i,j^{th} entry of AC times the block BD, which is the i,j^{th} block of $AC \otimes BD$.

□

If particular, if $A, B \in \mathbb{C}^{m\times m}$ and $\mathbf{x}, \mathbf{y} \in \mathbb{C}^m$ then

$$(A \otimes B)(\mathbf{x} \otimes \mathbf{y}) = A\mathbf{x} \otimes B\mathbf{y} \tag{2.9}$$

We will use this property in the next section.

2.2 Derivation of the Kronecker product form

We now show that all solutions to (2.1) and (2.2) are solutions to (2.3) and (2.4).

The matrix equations (2.1) & (2.2) are equivalent to:

$$\begin{aligned} A_1\mathbf{x} - \lambda B_1\mathbf{x} - \mu C_1\mathbf{x} &= \mathbf{0} && (2.10)\\ A_2\mathbf{y} - \lambda B_2\mathbf{y} - \mu C_2\mathbf{y} &= \mathbf{0} && (2.11) \end{aligned}$$

So Kronecker multiplying (2.10) on the right by $C_2\mathbf{y}$ and (2.11) on the left by $C_1\mathbf{x}$ and equating we get:

$$\begin{aligned} & (A_1\mathbf{x} - \lambda B_1\mathbf{x} - \mu C_1\mathbf{x}) \otimes C_2\mathbf{y} &&= C_1\mathbf{x} \otimes (A_2\mathbf{y} - \lambda B_2\mathbf{y} - \mu C_2\mathbf{y})\\ \Rightarrow\quad & (A_1\mathbf{x} - \lambda B_1\mathbf{x}) \otimes C_2\mathbf{y} &&= C_1\mathbf{x} \otimes (A_2\mathbf{y} - \lambda B_2\mathbf{y})\\ \Rightarrow\quad & A_1\mathbf{x} \otimes C_2\mathbf{y} - \lambda(B_1\mathbf{x} \otimes C_2\mathbf{y}) &&= C_1\mathbf{x} \otimes A_2\mathbf{y} - \lambda(C_1\mathbf{x} \otimes B_2\mathbf{y})\\ \Rightarrow\quad & A_1\mathbf{x} \otimes C_2\mathbf{y} - C_1\mathbf{x} \otimes A_2\mathbf{y} &&= \lambda(B_1\mathbf{x} \otimes C_2\mathbf{y} - C_1\mathbf{x} \otimes B_2\mathbf{y})\\ \Rightarrow\quad & (A_1 \otimes C_2 - C_1 \otimes A_2)(\mathbf{x} \otimes \mathbf{y}) &&= \lambda(B_1 \otimes C_2 - C_1 \otimes B_2)(\mathbf{x} \otimes \mathbf{y})\\ \Rightarrow\quad & \Delta_1\mathbf{z} &&= \lambda\Delta_0\mathbf{z}, \qquad (2.12) \end{aligned}$$

where in going from the antepenultimate line to the penultimate line we have used (2.9). Similarly, Kronecker multiplying (2.10) on the right by $B_2\mathbf{y}$ and (2.11) on the left by $B_1\mathbf{x}$ we get:

$$\begin{aligned} & (A_1\mathbf{x} - \lambda B_1\mathbf{x} - \mu C_1\mathbf{x}) \otimes B_2\mathbf{y} &&= B_1\mathbf{x} \otimes (A_2\mathbf{y} - \lambda B_2\mathbf{y} - \mu B_2\mathbf{y})\\ \Rightarrow\quad & (A_1\mathbf{x} - \mu C_1\mathbf{x}) \otimes B_2\mathbf{y} &&= B_1\mathbf{x} \otimes (A_2\mathbf{y} - \mu C_2\mathbf{y})\\ \Rightarrow\quad & B_1\mathbf{x} \otimes A_2\mathbf{y} - A_1\mathbf{x} \otimes B_2\mathbf{y} &&= \mu(B_1\mathbf{x} \otimes C_2\mathbf{y} - C_1\mathbf{x} \otimes B_2\mathbf{y})\\ \Rightarrow\quad & (B_1 \otimes A_2 - A_1 \otimes B_2)(\mathbf{x} \otimes \mathbf{y}) &&= \mu(B_1 \otimes C_2 - C_1 \otimes B_2)(\mathbf{x} \otimes \mathbf{y})\\ \Rightarrow\quad & \Delta_2\mathbf{z} &&= \mu\Delta_0\mathbf{z} \qquad (2.13) \end{aligned}$$

Hence we have the required forms (2.3) & (2.4), and so any solution to (2.1) & (2.2) will correspond to a solution of (2.3) & (2.4).

The advantage of applying to Kronecker product here is that we have now separated out the equations for λ and μ into two separate generalised eigenvalue problems. These can be solved using any number of common techniques to find the eigenvalues and eigenvectors of either equation (2.3) or (2.4). The eigenvectors can then be transfered to the remaining equation in order to check which also are eigenvectors of that equation, as an eigenvector of the whole system must satisfy both (2.3) and (2.4). The major drawback to this technique is the significant increase in the dimension of the problem. If we are in the situation where both sets of matrices A_i, B_i and C_i have dimension $n \times n$, then the dimension of the corresponding Kronecker system is of dimension $n^2 \times n^2$. It is this increase in complexity that has driven the development of other numerical techniques for solving two parameter eigenvalue problems that do not employ the Kronecker form (2.3) & (2.4).

2.3 Construction of a model problem

In order to test any methods for solving a two parameter eigenvalue problem we first will need one in which we know all its eigenvalues and eigenvectors explicitly. In this section we shall build such a system from diagonal matrices as these will give us some simple simultaneous equations to be solved.

Consider the following system of equations:

$$\begin{bmatrix}1 & 0\\ 0 & 2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} = \lambda\begin{bmatrix}3 & 0\\ 0 & 4\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \mu\begin{bmatrix}5 & 0\\ 0 & 6\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} \tag{2.14}$$

$$\begin{bmatrix}2 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}y_1\\ y_2\end{bmatrix} = \lambda\begin{bmatrix}8 & 0\\ 0 & 9\end{bmatrix}\begin{bmatrix}y_1\\ y_2\end{bmatrix} + \mu\begin{bmatrix}10 & 0\\ 0 & 11\end{bmatrix}\begin{bmatrix}y_1\\ y_2\end{bmatrix} \tag{2.15}$$

Due to the diagonal nature of these matrices we get the 4 linear equations:

$$\begin{aligned} x_1 &= (3\lambda + 5\mu)x_1\\ 2x_2 &= (4\lambda + 6\mu)x_2\\ 2y_1 &= (8\lambda + 10\mu)y_1\\ y_2 &= (9\lambda + 11\mu)y_2 \end{aligned}$$

So

$$x_1 \neq 0 \quad \Rightarrow \quad 3\lambda + 5\mu = 1 \tag{2.16}$$

$$x_2 \neq 0 \quad \Rightarrow \quad 2\lambda + 3\mu = 1 \tag{2.17}$$

$$y_1 \neq 0 \quad \Rightarrow \quad 4\lambda + 5\mu = 1 \tag{2.18}$$

$$y_2 \neq 0 \quad \Rightarrow \quad 9\lambda + 11\mu = 1 \tag{2.19}$$

Note $x_i \neq 0 \Rightarrow x_j = 0$, for $i \neq j$ else both $y_1 = 0$ and $y_2 = 0$ and so we would have $\mathbf{y} = \mathbf{0}$.
Similarly $y_i \neq 0 \Rightarrow y_j = 0$, for $i \neq j$.

Now noting that an eigenvector can be scaled by an arbitrary nonzero scalar, we set, without loss of generality, x_i and $y_i = 1$ in the corresponding eigenvector. Hence we have 4 pairs of eigenvectors:

$$\mathbf{x}, \mathbf{y} = \begin{bmatrix}1\\ 0\end{bmatrix}, \begin{bmatrix}1\\ 0\end{bmatrix} \tag{2.20}$$

$$\mathbf{x}, \mathbf{y} = \begin{bmatrix}1\\ 0\end{bmatrix}, \begin{bmatrix}0\\ 1\end{bmatrix} \tag{2.21}$$

$$\mathbf{x}, \mathbf{y} = \begin{bmatrix}0\\ 1\end{bmatrix}, \begin{bmatrix}0\\ 1\end{bmatrix} \tag{2.22}$$

$$\mathbf{x}, \mathbf{y} = \begin{bmatrix}0\\ 1\end{bmatrix}, \begin{bmatrix}1\\ 0\end{bmatrix} \tag{2.23}$$

So in case (2.20) we have the pair of simultaneous equations (2.16) and (2.18). Hence these imply that $\lambda = 0, \mu = 1/5$.

In case (2.21) equations (2.16) & (2.19) correspond to an eigenvalue $(\lambda, \mu) = (-1/2, 1/2)$. In case (2.22) equations (2.17) & (2.19) give rise to an eigenvalue $(\lambda, \mu) = (-8/5, 7/5)$ and in (2.23) the simultaneous equations (2.17) & (2.18) correspond to eigenvalues $(\lambda, \mu) = (-1, 1)$.

2.3.1 Example of Kronecker product form

In this section we shall use the previously constructed model example from §2.3 and show what the corresponding Kronecker Product form is.

So in the matrix form of the two parameter eigenvalue problem if we take the following as our matrices:

$A1 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$, $B1 = \begin{bmatrix} 3 & 0 \\ 0 & 4 \end{bmatrix}$, $C1 = \begin{bmatrix} 5 & 0 \\ 0 & 6 \end{bmatrix}$,

$A2 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$, $B2 = \begin{bmatrix} 8 & 0 \\ 0 & 9 \end{bmatrix}$, $C2 = \begin{bmatrix} 10 & 0 \\ 0 & 11 \end{bmatrix}$.

So using the formula for Δ_0 given by equation (2.5) we have

$$\begin{aligned} \Delta_0 &= B_1 \otimes C_2 - C_1 \otimes B_2 \\ &= \begin{bmatrix} 3 & 0 \\ 0 & 4 \end{bmatrix} \otimes \begin{bmatrix} 10 & 0 \\ 0 & 11 \end{bmatrix} - \begin{bmatrix} 5 & 0 \\ 0 & 6 \end{bmatrix} \otimes \begin{bmatrix} 8 & 0 \\ 0 & 9 \end{bmatrix} \\ &= \begin{bmatrix} 30 & 0 & 0 & 0 \\ 0 & 33 & 0 & 0 \\ 0 & 0 & 40 & 0 \\ 0 & 0 & 0 & 44 \end{bmatrix} - \begin{bmatrix} 40 & 0 & 0 & 0 \\ 0 & 45 & 0 & 0 \\ 0 & 0 & 48 & 0 \\ 0 & 0 & 0 & 54 \end{bmatrix} \\ &= \begin{bmatrix} -10 & 0 & 0 & 0 \\ 0 & -12 & 0 & 0 \\ 0 & 0 & -8 & 0 \\ 0 & 0 & 0 & -10 \end{bmatrix} \end{aligned} \tag{2.24}$$

Similarly, using (2.6) we get

$$\begin{aligned} \Delta_1 &= A_1 \otimes C_2 - C_1 \otimes A_2 \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \otimes \begin{bmatrix} 10 & 0 \\ 0 & 11 \end{bmatrix} - \begin{bmatrix} 5 & 0 \\ 0 & 6 \end{bmatrix} \otimes \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 11 & 0 & 0 \\ 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 22 \end{bmatrix} - \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 12 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 16 \end{bmatrix} \end{aligned} \tag{2.25}$$

And using (2.7) we get

$$\begin{aligned}\Delta_2 &= B_1 \otimes A_2 - A_1 \otimes B_2 \\ &= \begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & -6 & 0 & 0 \\ 0 & 0 & -8 & 0 \\ 0 & 0 & 0 & -14 \end{bmatrix}\end{aligned} \tag{2.26}$$

Hence the Kronecker system of the two parameter eigenvalue problem is

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 16 \end{bmatrix} \mathbf{z} = \lambda \begin{bmatrix} -10 & 0 & 0 & 0 \\ 0 & -12 & 0 & 0 \\ 0 & 0 & -8 & 0 \\ 0 & 0 & 0 & -10 \end{bmatrix} \mathbf{z} \tag{2.27}$$

$$\begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & -6 & 0 & 0 \\ 0 & 0 & -8 & 0 \\ 0 & 0 & 0 & -14 \end{bmatrix} \mathbf{z} = \mu \begin{bmatrix} -10 & 0 & 0 & 0 \\ 0 & -12 & 0 & 0 \\ 0 & 0 & -8 & 0 \\ 0 & 0 & 0 & -10 \end{bmatrix} \mathbf{z} \tag{2.28}$$

Now if we turn our attention to the eigenvectors of the system in §2.3 we can calculate the corresponding eigenvectors of the Kronecker system.

(2.20) corresponds to the eigenvector $\mathbf{z} = [1, 0, 0, 0]^T$ and eigenvalue $(0, \frac{1}{5})$. (2.21) corresponds to the eigenvector $\mathbf{z} = [0, 1, 0, 0]^T$ and eigenvalue $(-\frac{1}{2}, \frac{1}{2})$. (2.22) corresponds to the eigenvector $\mathbf{z} = [0, 0, 0, 1]^T$ and eigenvalue $(-\frac{8}{5}, \frac{7}{5})$, and (2.23) corresponds to the eigenvector $\mathbf{z} = [0, 0, 1, 0]^T$ and eigenvalue $(-1, 1)$. Now we note that the eigenvectors here correspond exactly with the eigenvalues given in the model problem.

Chapter 3

Implementation of Newton's method

In this chapter we will show how Newton's method can be applied to a two parameter eigenvalue problem. We will also show that, for simple eigenvalues, classical properties of Newton's method still occur, notably quadratic convergence.

3.1 Theory of Newton's method

In order to apply Newton's method to a two parameter eigenvalue problem we must first recap the standard Newton's method. Consider the following system to be solved

$$\mathbf{F}(\mathbf{x}) = \mathbf{0} \tag{3.1}$$

The algorithm for implementing Newton's method is then the following:
Begin with an initial approximation $\mathbf{x}^0$ for the solution. Then for $k \in \mathbb{Z}^+$

1. Solve the system
$$\mathbf{F}_{\mathbf{x}}(\mathbf{x}^k)\mathbf{d}^k = -\mathbf{F}(\mathbf{x}^k) \tag{3.2}$$
2. Update the approximation by
$$\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{d}^k \tag{3.3}$$

Now for Newton's method to be applied, we can see that the solve (3.2) can only be computed if the Jacobian matrix $\mathbf{F}_{\mathbf{x}}(\mathbf{x}^k)$ is nonsingular. More specifically, if the Jacobian at the root is nonsingular, then the Jacobian will be nonsingular for a guess $\mathbf{x}^k$ sufficiently close to the root $\mathbf{x}^*$, by standard Newton convergence theory discussed in [4].

Now for a two parameter eigenvalue problem we have to solve the following system

$$\mathbf{F(v)} = \mathbf{0}$$

where

$$\mathbf{F}(\mathbf{v}) = \begin{bmatrix} A_1\mathbf{x} - \lambda B_1\mathbf{x} - \mu C_1\mathbf{x} \\ -\frac{1}{2}\mathbf{x}^T\mathbf{x} + \frac{1}{2} \\ A_2\mathbf{y} - \lambda B_2\mathbf{y} - \mu C_2\mathbf{y} \\ -\frac{1}{2}\mathbf{y}^T\mathbf{y} + \frac{1}{2} \end{bmatrix} \tag{3.4}$$

and

$$\mathbf{v} = [\mathbf{x}, \lambda, \mathbf{y}, \mu]^T \in \mathbb{C}^{n+m+2} \tag{3.5}$$

is a root of $\mathbf{F}$

From now on we shall consider only two parameter eigenvalue problems with real coefficients. This said, it is highly likely that the theory will still be applicable in the complex world where the relevant changes are made. Also we shall consider systems of two parameter eigenvalue problems where the dimension of (2.1) and (2.2) are the same, i.e. $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. Hence $\mathbf{v} \in \mathbb{R}^{2n+2}$ and $\mathbf{F}_\mathbf{v}(\mathbf{v}) \in \mathbb{R}^{(2n+2)\times(2n+2)}$.

3.2 Validity of Newton's method for two parameter eigenvalue problems

In this section we wish to show when Newton's method is well-defined, i.e. a certain Jacobian is nonsingular. Let us begin by stating some results that we will use and make some important definitions.

Lemma 3.2.1. *(ABCD Lemma)[4] Given an $n \times n$ matrix A with $rank(A) = (n-1)$ and $\boldsymbol{b}, \boldsymbol{c} \in \mathbb{C}^n$ and $d \in \mathbb{C}$. Consider the bordered $(n+1)\times(n+1)$ matrix*

$$M = \begin{pmatrix} A & \boldsymbol{b} \\ \boldsymbol{c}^T & d \end{pmatrix} \tag{3.6}$$

Then M is nonsingular if and only if

$$\psi^T\boldsymbol{b} \neq 0 \qquad \forall\psi \in Ker(A^T)\backslash\{\boldsymbol{0}\} \tag{3.7}$$

and

$$\boldsymbol{c}^T\theta \neq 0 \qquad \forall\theta \in Ker(A)\backslash\{\boldsymbol{0}\} \tag{3.8}$$

The proof of this lemma is simple and can be found in [4].

Definition 3.2.1. *An eigenvalue μ satisfying $A\boldsymbol{x} = \mu\boldsymbol{x}$ is said to be (geometrically)* simple *if and only if $dimKer(A - \mu I) = 1$.*

Definition 3.2.2. *An eigenvalue (λ, μ) of the two parameter eigenvalue problem satisfying (2.1) and (2.2) is said to be (geometrically)* simple *if and only if $dimKer(A_1 - \lambda B_1 - \mu C_1) = 1$ and $dimKer(A_2 - \lambda B_2 - \mu C_2) = 1$.*

Definition 3.2.3. *An eigenvector is said to be* simple *if it corresponds to a simple eigenvalue.*

From now on, a simple eigenvalue shall refer to a geometrically simple eigenvalue.

For Newton's method applied to $\mathbf{F}(\mathbf{v}) = \mathbf{0}$ given by (3.4) we have the Jacobian matrix

$$\mathbf{F}_{\mathbf{v}}(\mathbf{v}) = \begin{bmatrix} A_1 - \lambda B_1 - \mu C_1 & -B_1\mathbf{x} & O & -C_1\mathbf{x} \\ -\mathbf{x}^T & 0 & \mathbf{0}^T & 0 \\ O & -B_2\mathbf{y} & A_2 - \lambda B_2 - \mu C_2 & -C_2\mathbf{y} \\ \mathbf{0}^T & 0 & -\mathbf{y}^T & 0 \end{bmatrix} \tag{3.9}$$

Our goal is to provide a simple condition that, if satisfied, will ensure that Newton's method is well-defined. More specifically, we wish to find a sufficient condition for the Jacobian $\mathbf{F}_{\mathbf{v}}(\mathbf{v})$ to be nonsingular. We shall do this in a series of lemmata, the first of which is a condition that the $(n+1) \times (n+1)$ matrix in the top left corner of the Jacobian be nonsingular.

Let us first introduce some notation.

$$\mathbf{P}^{(1)} := \begin{bmatrix} A_1 - \lambda B_1 - \mu C_1 & -B_1\mathbf{x} \\ -\mathbf{x}^T & 0 \end{bmatrix} \tag{3.10}$$

$$\mathbf{P}^{(2)} := \begin{bmatrix} O & -C_1\mathbf{x} \\ \mathbf{0}^T & 0 \end{bmatrix} \tag{3.11}$$

$$\mathbf{P}^{(3)} := \begin{bmatrix} O & -B_2\mathbf{y} \\ \mathbf{0}^T & 0 \end{bmatrix} \tag{3.12}$$

$$\mathbf{P}^{(1)} := \begin{bmatrix} A_2 - \lambda B_2 - \mu C_2 & -C_2\mathbf{y} \\ -\mathbf{y}^T & 0 \end{bmatrix} \tag{3.13}$$

So that $\mathbf{F}_{\mathbf{v}}(\mathbf{v}) = \begin{bmatrix} \mathbf{P}^{(1)} & \mathbf{P}^{(2)} \\ \mathbf{P}^{(3)} & \mathbf{P}^{(4)} \end{bmatrix}$.

Lemma 3.2.2. *Assuming that $(\boldsymbol{x}, \lambda, \boldsymbol{y}, \mu)$ is a simple eigenpair for (2.1) and (2.2) then*

$$Rank\left[A_i - \lambda B_i - \mu C_i\right] = n - 1 \qquad i = 1, 2 \tag{3.14}$$

where n is the dimension of the matrix $\left[A_i - \lambda B_i - \mu C_i\right]$.

Moreover if

$$\psi^T B_1 \boldsymbol{x} \neq 0 \qquad \forall \psi \in Ker([A_1 - \lambda B_1 - \mu C_1]^T) \backslash \{\boldsymbol{0}\} \tag{3.15}$$

then $\mathbf{P}^{(1)} = \begin{bmatrix} A_1 - \lambda B_1 - \mu C_1 & -B_1\boldsymbol{x} \\ -\boldsymbol{x}^T & 0 \end{bmatrix}$ is nonsingular.

Proof. We prove (3.14) using the Rank-Nullity Theorem from linear algebra. As $(\mathbf{x}, \lambda, \mathbf{y}, \mu)$ is a simple eigenpair, this means that $dim Ker(A_i - \lambda B_i - \mu C_i) = 1$.

The Rank-Nullity Theorem states

$$Rank(A_i - \lambda B_i - \mu C_i) + dim Ker(A_i - \lambda B_i - \mu C_i) = n \tag{3.16}$$

where n is the number of columns of $(A_i - \lambda B_i - \mu C_i)$. From this it is immediate that $Rank(A_i - \lambda B_i - \mu C_i) = n - 1$, for $i = 1, 2$.

For the second part of the lemma we have

$$\mathbf{P}^{(1)} = \begin{bmatrix} A_1 - \lambda B_1 - \mu C_1 & -B_1\mathbf{x} \\ -\mathbf{x}^T & 0 \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)} \tag{3.17}$$

By the ABCD Lemma we have $\mathbf{P}^{(1)}$ nonsingular if and only if

$$\psi^T B_1 \mathbf{x} \neq 0 \quad \forall \psi \in Ker([A_1 - \lambda B_1 - \mu C_1]^T) \backslash \{\mathbf{0}\} \tag{3.18}$$

$$\mathbf{x}^T \theta \neq 0 \quad \forall \theta \in Ker(A_1 - \lambda B_1 - \mu C_1) \backslash \{\mathbf{0}\}. \tag{3.19}$$

We have assumed, (λ, μ) is a simple eigenvalue, this implies that $\mathbf{x}$ is the only corresponding nontrivial right eigenvector of $A_1 - \lambda B_1 - \mu C_1$. Hence

$$\mathbf{x}^T \theta = \mathbf{x}^T \mathbf{x} \neq 0 \qquad \forall \theta \in Ker(A_1 - \lambda B_1 - \mu C_1) \backslash \{\mathbf{0}\}. \tag{3.20}$$

So condition (3.19) is always satisfied and hence $\mathbf{P}^{(1)}$ nonsingular if and only if $\psi^T B_1 \mathbf{x} \neq 0 \qquad \forall \psi \in Ker([A_1 - \lambda B_1 - \mu C_1]^T) \backslash \{\mathbf{0}\}$.

□

Now we make use of the structure of the Jacobian of $\mathbf{F}$ to learn more about it by performing a block LU factorisation of $\mathbf{F_v}(\mathbf{v})$.

Lemma 3.2.3. *Assume we have a simple root $(\boldsymbol{x}, \lambda, \boldsymbol{y}, \mu)$ of $\boldsymbol{F}$ and,*

$$\psi^T B_1 \boldsymbol{x} \neq 0 \tag{3.21}$$

where ψ satisfies $\psi \in Ker([A_1 - \lambda B_1 - \mu C_1]^T) \backslash \{\boldsymbol{0}\}$,
and

$$\frac{1}{\psi^T B_1 \boldsymbol{x}} \begin{vmatrix} \phi^T C_2 \boldsymbol{y} & \phi^T B_2 \boldsymbol{y} \\ \psi^T C_1 \boldsymbol{x} & \psi^T B_1 \boldsymbol{x} \end{vmatrix} \neq 0. \tag{3.22}$$

Then the Jacobian $\boldsymbol{F_v}(\boldsymbol{v})$ is nonsingular, where ϕ satisfies
$\phi \in Ker([A_2 - \lambda B_2 - \mu C_2]^T) \backslash \{\boldsymbol{0}\}$.

Proof. Consider the $(n+1) \times (n+1)$ matrix in the top left corner of $\mathbf{F_v}(\mathbf{v})$: Now with (3.21) we can apply (Lemma 3.2.2) to see that $\mathbf{P}^{(1)}$ is nonsingular, and in particular $(\mathbf{P}^{(1)})^{-1}$ exists.

Writing $\mathbf{F_v}(\mathbf{v})$ as $\begin{bmatrix} \mathbf{P}^{(1)} & \mathbf{P}^{(2)} \\ \mathbf{P}^{(3)} & \mathbf{P}^{(4)} \end{bmatrix}$ where $\mathbf{P}^{(i)} \in \mathbb{R}^{(n+1)\times(n+1)}$ we can perform a block LU factorisation:

$$\mathbf{F_v}(\mathbf{v}) = \begin{bmatrix} \mathbf{P}^{(1)} & \mathbf{P}^{(2)} \\ \mathbf{P}^{(3)} & \mathbf{P}^{(4)} \end{bmatrix} = \begin{bmatrix} I & O \\ \mathbf{P}^{(3)}(\mathbf{P}^{(1)})^{-1} & I \end{bmatrix} \begin{bmatrix} \mathbf{P}^{(1)} & \mathbf{P}^{(2)} \\ O & \mathbf{P}^{(4)} - \mathbf{P}^{(3)}(\mathbf{P}^{(1)})^{-1}\mathbf{P}^{(2)} \end{bmatrix} \tag{3.23}$$

Thus, $\mathbf{F_v}(\mathbf{v})$ is nonsingular if and only if $[\mathbf{P}^{(4)} - \mathbf{P}^{(3)}(\mathbf{P}^{(1)})^{-1}\mathbf{P}^{(2)}]$ is nonsingular.

Consider $\mathbf{Q} = (\mathbf{P}^{(1)})^{-1}\mathbf{P}^{(2)} \Leftrightarrow \mathbf{P}^{(2)} = \mathbf{P}^{(1)}\mathbf{Q}$. So $\mathbf{F_v}(\mathbf{v})$ is nonsingular if and only if $\mathbf{P}^{(4)} - \mathbf{P}^{(3)}\mathbf{Q}$ is nonsingular. To find Q consider the system $\mathbf{P}^{(1)}\mathbf{Q} = \mathbf{P}^{(2)}$ where $\mathbf{Q}$ is unknown:

$$\begin{bmatrix} A_1 - \lambda B_1 - \mu C_1 & -B_1\mathbf{x} \\ -\mathbf{x}^T & 0 \end{bmatrix} \begin{bmatrix} * & \mathbf{k}_1 \\ *^T & k_2 \end{bmatrix} = \begin{bmatrix} O & -C_1\mathbf{x} \\ \mathbf{0}^T & 0 \end{bmatrix} \tag{3.24}$$

Thus $\mathbf{Q} = \begin{bmatrix} O & \mathbf{k}_1 \\ \mathbf{0}^T & k_2 \end{bmatrix}$ for some $\mathbf{k}_1$ and k_2. However, as we need only know $\mathbf{P}^{(3)}\mathbf{Q}$, we make use of the sparse nature of $\mathbf{P}^{(3)}$:

$$\mathbf{P}^{(3)} = \begin{bmatrix} O & -B_2\mathbf{y} \\ \mathbf{0}^T & 0 \end{bmatrix}$$

and realise that only k_2 is needed. So from (3.24) we have

$$[A_1 - \lambda B_1 - \mu C_1]\mathbf{k}_1 - k_2 B_1\mathbf{x} = -C_1\mathbf{x}$$

Multiplying on the left by ψ^T, where $\psi \in \text{Ker}([A_1 - \lambda B_1 - \mu C_1]^T)\backslash\{\mathbf{0}\}$ gives

$$\begin{aligned} \psi^T[A_1 - \lambda B_1 - \mu C_1]\mathbf{k}_1 - \psi^T B_1\mathbf{x}k_2 &= -\psi^T C_1\mathbf{x} \\ \Leftrightarrow \qquad -\psi^T B_1\mathbf{x}k_2 &= -\psi^T C_1\mathbf{x}. \end{aligned}$$

By assumption we see that $\psi^T B_1\mathbf{x} \neq 0$, hence

$$k_2 = \frac{\psi^T C_1\mathbf{x}}{\psi^T B_1\mathbf{x}} \tag{3.25}$$

Thus $\mathbf{P}^{(3)}(\mathbf{P}^{(1)})^{-1}\mathbf{P}^{(2)}$ is

$$\begin{bmatrix} O & -B_2\mathbf{y} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} A_1 - \lambda B_1 - \mu C_1 & -B_1\mathbf{x} \\ -\mathbf{x}^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} O & -C_1\mathbf{x} \\ \mathbf{0}^T & 0 \end{bmatrix} = \begin{bmatrix} O & -k_2 B_2\mathbf{y} \\ \mathbf{0}^T & 0 \end{bmatrix}$$

So $[\mathbf{P}^{(4)} - \mathbf{P}^{(3)}(\mathbf{P}^{(1)})^{-1}\mathbf{P}^{(2)}]$ equals

$$\begin{bmatrix} A_2 - \lambda B_2 - \mu C_2 & -C_2\mathbf{y} + k_2 B_2\mathbf{y} \\ -\mathbf{y}^T & 0 \end{bmatrix} = \begin{bmatrix} A_2 - \lambda B_2 - \mu C_2 & -(C_2 - k_2 B_2)\mathbf{y} \\ -\mathbf{y}^T & 0 \end{bmatrix} \tag{3.26}$$

Applying the ABCD lemma to this matrix we find that (3.26) is nonsingular if

$$\phi^T(C_2 - k_2 B_2)\mathbf{y} \neq 0 \qquad \forall \phi \in Ker([A_2 - \lambda B_2 - \mu C_2]^T)\backslash\{\mathbf{0}\} \tag{3.27}$$

noting that $\mathbf{y}$ is the only right eigenvector of $[A_2 - \lambda B_2 - \mu C_2]$.

So $\begin{bmatrix} \mathbf{P}^{(1)} & \mathbf{P}^{(2)} \\ \mathbf{P}^{(3)} & \mathbf{P}^{(4)} \end{bmatrix}$ is nonsingular if

$$\psi^T B_1\mathbf{x} \neq 0 \qquad \forall \psi \in Ker([A_1 - \lambda B_1 - \mu C_1]^T)\backslash\{\mathbf{0}\} \tag{3.28}$$

and

$$\phi^T(C_2 - \frac{\psi^T C_1 \mathbf{x}}{\psi^T B_1 \mathbf{x}} B_2)\mathbf{y} \neq 0 \qquad \forall \phi \in Ker([A_2 - \lambda B_2 - \mu C_2]^T)\backslash\{\mathbf{0}\} \quad (3.29)$$

or equivalently

$$\frac{1}{\psi^T B_1 \mathbf{x}} \begin{vmatrix} \phi^T C_2 \mathbf{y} & \phi^T B_2 \mathbf{y} \\ \psi^T C_1 \mathbf{x} & \psi^T B_1 \mathbf{x} \end{vmatrix} \neq 0 \quad (3.30)$$

□

However, (3.21) & (3.22) are not the only conditions that ensure $\mathbf{F_v}(\mathbf{v})$ is nonsingular, as we show in the next section.

3.3 Derivation of more conditions for a nonsingular Jacobian

Now at the start of (Lemma 3.2.3) we performed a block LU factorisation of the Jacobian $\mathbf{F_v}(\mathbf{v})$, ie we decomposed the matrix as follows

$$\begin{bmatrix} * & \\ * & * \end{bmatrix} \begin{bmatrix} * & * \\ & * \end{bmatrix} = \mathbf{F_v}(\mathbf{v}) \quad (3.31)$$

If instead we factored the Jacobian into a block upper triangular matrix multiplied on the right by a block lower triangular matrix we get a different condition for the Jacobian being nonsingular, i.e. we decompose the Jacobian in the following way

$$\begin{bmatrix} * & * \\ & * \end{bmatrix} \begin{bmatrix} * & \\ * & * \end{bmatrix} = \mathbf{F_v}(\mathbf{v}) \quad (3.32)$$

Lemma 3.3.1. *Assume we have a simple root $(\lambda, \mu, \boldsymbol{x}, \boldsymbol{y})$ of $\boldsymbol{F}$ and, if*

$$\phi^T C_2 \boldsymbol{y} \neq 0 \qquad \forall \phi \in Ker([A_2 - \lambda B_2 - \mu C_2]^T)\backslash\{\boldsymbol{0}\} \quad (3.33)$$

and

$$\frac{1}{\phi^T C_2 \boldsymbol{y}} \begin{vmatrix} \phi^T C_2 \boldsymbol{y} & \phi^T B_2 \boldsymbol{y} \\ \psi^T C_1 \boldsymbol{x} & \psi^T B_1 \boldsymbol{x} \end{vmatrix} \neq 0 \quad (3.34)$$

$\forall \psi \in Ker([A_1 - \lambda B_1 - \mu C_1]^T)\backslash\{\boldsymbol{0}\}$

then the Jacobian $\boldsymbol{F_v}(\boldsymbol{v})$ is nonsingular.

Proof. As in the proof of (Lemma 3.2.3), assuming (3.33) we can apply (Lemma 3.2.2) to see that $\mathbf{P}^{(4)}$ is nonsingular, and in particular $(\mathbf{P}^{(4)})^{-1}$ exists.

Now write the Jacobian of $\mathbf{F}$ as

$$\mathbf{F_v}(\mathbf{v}) = \begin{bmatrix} \mathbf{P}^{(1)} & \mathbf{P}^{(2)} \\ \mathbf{P}^{(3)} & \mathbf{P}^{(4)} \end{bmatrix} = \begin{bmatrix} I & \mathbf{P}^{(2)}(\mathbf{P}^{(4)})^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{P}^{(1)} - \mathbf{P}^{(2)}(\mathbf{P}^{(4)})^{-1}\mathbf{P}^{(3)} & 0 \\ \mathbf{P}^{(3)} & \mathbf{P}^{(4)} \end{bmatrix} \quad (3.35)$$

So $\mathbf{F_v}(\mathbf{v})$ is nonsingular if $det[\mathbf{P}^{(1)} - \mathbf{P}^{(2)}(\mathbf{P}^{(4)})^{-1}\mathbf{P}^{(3)}] \neq 0$.

Applying the same arguments as in the proof of (Lemma 3.2.3) as to the sparse structure of $\mathbf{P}^{(2)}$ and $\mathbf{P}^{(3)}$ we conclude

$$\mathbf{P}^{(2)}(\mathbf{P}^{(4)})^{-1}\mathbf{P}^{(3)} = \begin{bmatrix} O & -\frac{\phi^T B_2 \mathbf{y}}{\phi^T C_2 \mathbf{y}} C_1 \mathbf{x} \\ \mathbf{0}^T & 0 \end{bmatrix} \tag{3.36}$$

For all $\phi \in Ker([A_2 - \lambda B_2 - \mu C_2]^T)\backslash\{\mathbf{0}\}$. Hence

$$\mathbf{P}^{(1)} - \mathbf{P}^{(2)}(\mathbf{P}^{(4)})^{-1}\mathbf{P}^{(3)} = \begin{bmatrix} A_1 - \lambda B_1 - \mu C_1 & -(B_1 - \frac{\phi^T B_2 \mathbf{y}}{\phi^T C_2 \mathbf{y}} C_1)\mathbf{x} \\ \mathbf{x}^T & 0 \end{bmatrix} \tag{3.37}$$

So applying the *ABCD Lemma* to this matrix, we get that it is nonsingular if and only if

$$\psi^T (B_1 - \frac{\phi^T B_2 \mathbf{y}}{\phi^T C_2 \mathbf{y}} C_1)\mathbf{x} \neq 0 \qquad \forall \psi \in Ker([A_1 - \lambda B_1 - \mu C_1]^T)\backslash\{\mathbf{0}\} \tag{3.38}$$

And now we note that (3.38) is equivalent to (3.34).

□

Going back to our definition of $\mathbf{v}$ and, in particular, the order in which it was written we can see that the choice $\mathbf{v} = [\mathbf{x}, \lambda, \mathbf{y}, \mu]^T$ was only made in order to split the Jacobian up into square block matrices. We could equally have written

$$\mathbf{v} = [\mathbf{x}, \mu, \mathbf{y}, \lambda]^T$$

Thus we then have

$$\mathbf{F}_{\mathbf{v}}(\mathbf{v}) = \begin{bmatrix} A_1 - \lambda B_1 - \mu C_1 & -C_1\mathbf{x} & O & -B_1\mathbf{x} \\ -\mathbf{x}^T & 0 & \mathbf{0}^T & 0 \\ O & -C_2\mathbf{y} & A_2 - \lambda B_2 - \mu C_2 & -B_2\mathbf{y} \\ \mathbf{0}^T & 0 & -\mathbf{y}^T & 0 \end{bmatrix} \tag{3.39}$$

If this new ordering is used then we have the following Lemma.

Lemma 3.3.2. *Assume we have a simple root $(\boldsymbol{x}, \lambda, \boldsymbol{y}, \mu)$ of $\boldsymbol{F}$. Then $\boldsymbol{F}_{\boldsymbol{v}}(\boldsymbol{v})$ is nonsingular if either*

$$\psi^T C_1 \boldsymbol{x} \neq 0 \quad \forall \psi \in Ker([A_1 - \lambda B_1 - \mu C_1]^T)\backslash\{\boldsymbol{0}\} \tag{3.40}$$

$$\textit{and} \quad \phi^T (B_2 - \frac{\psi^T B_1 \boldsymbol{x}}{\psi^T C_1 \boldsymbol{x}} C_2)\boldsymbol{y} \neq 0 \quad \forall \phi \in Ker([A_2 - \lambda B_2 - \mu C_2]^T)\backslash\{\boldsymbol{0}\} \tag{3.41}$$

or

$$\phi^T B_2 \boldsymbol{y} \neq 0 \quad \forall \phi \in Ker([A_2 - \lambda B_2 - \mu C_2]^T)\backslash\{\boldsymbol{0}\} \tag{3.42}$$

$$\textit{and} \quad \psi^T (C_1 - \frac{\phi^T C_2 \boldsymbol{y}}{\phi^T B_2 \boldsymbol{y}} B_1)\boldsymbol{x} \neq 0 \quad \forall \psi \in Ker([A_1 - \lambda B_1 - \mu C_1]^T)\backslash\{\boldsymbol{0}\} \tag{3.43}$$

Proof. The proof of these are similar to the proofs of (Lemma 3.2.3) and (Lemma 3.3.1).

□

With these lemmata we can now prove the main result of the chapter.

3.4 Condition for Newton's method to be applied

Using the lemmata proved in the previous section we can show that Newton's method is well-defined if one elegant condition holds. This is, in fact, that the determinant of a 2×2 matrix is nonzero as the following theorem details.

Theorem 3.4.1. *Assume we have a simple zero $(\boldsymbol{x}, \lambda, \boldsymbol{y}, \mu)$ of $\boldsymbol{F}$ given by (3.4) then the Jabobian matrix $\boldsymbol{F_v}(\boldsymbol{v})$ is nonsingular if*

$$\begin{vmatrix} \psi^T B_1 \boldsymbol{x} & \psi^T C_1 \boldsymbol{x} \\ \phi^T B_2 \boldsymbol{y} & \phi^T C_2 \boldsymbol{y} \end{vmatrix} \neq 0 \tag{3.44}$$

$$\forall \psi \in Ker([A_1 - \lambda B_1 - \mu C_1]^T) \backslash \{\boldsymbol{0}\} \tag{3.45}$$

$$\forall \phi \in Ker([A_2 - \lambda B_2 - \mu C_2]^T) \backslash \{\boldsymbol{0}\} \tag{3.46}$$

Proof. Let $\psi \in Ker([A_1 - \lambda B_1 - \mu C_1]^T) \backslash \{\mathbf{0}\}$ and
let $\phi \in Ker([A_2 - \lambda B_2 - \mu C_2]^T) \backslash \{\mathbf{0}\}$.

Firstly note that we can freely permute the rows and columns of the matrix $\begin{bmatrix} \psi^T B_1 \mathbf{x} & \psi^T C_1 \mathbf{x} \\ \phi^T B_2 \mathbf{y} & \phi^T C_2 \mathbf{y} \end{bmatrix}$, as this will only affect the sign of its determinant.

(Case 1) $\psi^T B_1 \mathbf{x} = 0$.
Then we must have both $\psi^T C_1 \mathbf{x} \neq 0$ and $\phi^T B_2 \mathbf{y} \neq 0$, else

$$Rank \begin{bmatrix} \psi^T B_1 \mathbf{x} & \psi^T C_1 \mathbf{x} \\ \phi^T B_2 \mathbf{y} & \phi^T C_2 \mathbf{y} \end{bmatrix} < 2$$

and its determinant would then equal zero. So then as $\psi^T C_1 \mathbf{x} \neq 0$ we have $\frac{1}{\psi^T C_1 \mathbf{x}} \begin{vmatrix} \psi^T B_1 \mathbf{x} & \psi^T C_1 \mathbf{x} \\ \phi^T B_2 \mathbf{y} & \phi^T C_2 \mathbf{y} \end{vmatrix} \neq 0$.

Hence we are in a position to apply (Lemma 3.3.2) and we deduce that $\mathbf{F_v}(\mathbf{v})$ is nonsingular.

(Case 2) $\phi^T C_2 \mathbf{y} = 0$.
Similarly in this case we must have both $\psi^T C_1 \mathbf{x} \neq 0$ and $\phi^T B_2 \mathbf{y} \neq 0$, else $Rank \begin{bmatrix} \psi^T B_1 \mathbf{x} & \psi^T C_1 \mathbf{x} \\ \phi^T B_2 \mathbf{y} & \phi^T C_2 \mathbf{y} \end{bmatrix} < 2$ and its determinant would then equal zero. Hence as in (Case 1) we have $\frac{1}{\psi^T C_1 \mathbf{x}} \begin{vmatrix} \psi^T B_1 \mathbf{x} & \psi^T C_1 \mathbf{x} \\ \phi^T B_2 \mathbf{y} & \phi^T C_2 \mathbf{y} \end{vmatrix} \neq 0$ and can again apply (Lemma 3.3.2) to deduce $\mathbf{F_v}(\mathbf{v})$ is nonsingular.

(Case 3) $\psi^T C_1 \mathbf{x} = 0$.
Here we must have $\psi^T B_1 \mathbf{x} \neq 0$ and $\phi^T C_2 \mathbf{y} \neq 0$ in order for the (3.44) to hold. Hence we are in a position to apply either (Lemma 3.2.3) or (Lemma 3.3.1) and so we deduce $\mathbf{F_v}(\mathbf{v})$ is nonsingular.

(Case 4) $\phi^T B_2 \mathbf{y} = 0$.
Here we must have $\psi^T B_1 \mathbf{x} \neq 0$ and $\phi^T C_2 \mathbf{y} \neq 0$ in order for the (3.44) to hold. We then see that we are in the same situation as in (Case 3) and we use either Lemma 3.2.3) or (Lemma 3.3.1) to deduce that $\mathbf{F_v}(\mathbf{v})$ is nonsingular.

So having (3.44) will allow us to apply one of the previous 3 lemmata, and so we conclude that (3.44) is sufficient for $\mathbf{F}_{\mathbf{v}}(\mathbf{v})$ to be nonsingular.

□

If we now consider the model problem constructed in §2.3 we can check whether or not Newton's method is well-defined for each of its eigenpairs. Firstly let us look at the zero at $(\mathbf{x}, \lambda, \mathbf{y}, \mu) = (\begin{bmatrix}1\\0\end{bmatrix}, 0, \begin{bmatrix}1\\0\end{bmatrix}, \frac{1}{5})$. We can see that

$$\begin{bmatrix}\psi^T B_1 \mathbf{x} & \psi^T C_1 \mathbf{x}\\ \phi^T B_2 \mathbf{y} & \phi^T C_2 \mathbf{y}\end{bmatrix} = \begin{bmatrix}3 & 5\\ 8 & 10\end{bmatrix}$$

So

$$\begin{vmatrix}\psi^T B_1 \mathbf{x} & \psi^T C_1 \mathbf{x}\\ \phi^T B_2 \mathbf{y} & \phi^T C_2 \mathbf{y}\end{vmatrix} = -10 \neq 0$$

and hence we can see that by Theorem 3.4.1 Newton's method is well-defined for a starting value close enough to this root.

Calculating the relevant determinants for the remaining 3 eigenpairs we find that none of them are equal to 0, and so in this simple diagonal example Newton's method is well-defined.

Chapter 4

Numerics for Newton's method

4.1 Derivation of a practical Newton's method

Newton's method for solving a system of equations $\mathbf{F(v) = 0}$ is the following:

1. Solve the system $\mathbf{F_v}(\mathbf{v}_k)\mathbf{d}_k = -\mathbf{F}(\mathbf{v}_k)$,

2. Update your solution $\mathbf{v}_{k+1} = \mathbf{v}_k + \mathbf{d}_k$.

So in our two parameter eigenvalue system the step (1) is of the form

$$\begin{bmatrix} A_1 - \lambda_k B_1 - \mu_k C_1 & O & -B_1\mathbf{x}_k & -C_1\mathbf{x}_k \\ O & A_2 - \lambda_k B_2 - \mu_k C_2 & -B_2\mathbf{y}_k & -C_2\mathbf{y}_k \\ \mathbf{x}^T & \mathbf{0}^T & 0 & 0 \\ \mathbf{0}^T & \mathbf{y}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x}_k \\ \Delta\mathbf{y}_k \\ \Delta\lambda_k \\ \Delta\mu_k \end{bmatrix} = \begin{bmatrix} -(A_1 - \lambda_k B_1 - \mu_k C_1)\mathbf{x}_k \\ -(A_2 - \lambda_k B_2 - \mu_k C_2)\mathbf{y}_k \\ \frac{1}{2}(1 - \mathbf{x}_k^T\mathbf{x}_k) \\ \frac{1}{2}(1 - \mathbf{y}_k^T\mathbf{y}_k) \end{bmatrix} \tag{4.1}$$

Where our $\mathbf{v}_k$ is of the form $\mathbf{v}_k = \begin{bmatrix}\mathbf{x}_k, \mathbf{y}_k, \lambda_k, \mu_k\end{bmatrix}^T$ and our $\mathbf{d}_k$ is of the form $\mathbf{d}_k = \begin{bmatrix}\Delta\mathbf{x}_k, \Delta\mathbf{y}_k, \Delta\lambda_k, \Delta\mu_k\end{bmatrix}^T$. Note that the ordering in $\mathbf{v}$ is different from in the previous section. Now if we tried solving this system directly using Gaussian elimination then the computational cost is $O((2n+2)^3) \approx O(8n^3)$. We can improve on this order by looking at each individual component of the above matrix system.

So consider the first component in (4.1). This gives the equation

$$(A_1 - \lambda_k B_1 - \mu_k C_1)\Delta\mathbf{x}_k - B_1\mathbf{x}_k\Delta\lambda_k - C_1\mathbf{x}_k\Delta\mu_k = -(A_1 - \lambda_k B_1 - \mu_k C_1)\mathbf{x}_k \quad (4.2)$$

$$\Leftrightarrow \quad (A_1 - \lambda_k B_1 - \mu_k C_1)(\mathbf{x}_k + \Delta\mathbf{x}_k) = B_1\mathbf{x}_k\Delta\lambda_k + C_1\mathbf{x}_k\Delta\mu_k \quad (4.3)$$

Similarly considering the second component we obtain the equation

$$(A_2 - \lambda_k B_2 - \mu_k C_2)(\mathbf{y}_k + \Delta\mathbf{y}_k) = B_2\mathbf{y}_k\Delta\lambda_k + C_2\mathbf{y}_k\Delta\mu_k \quad (4.4)$$

Considering the third equation we have

$$\mathbf{x}_k^T\Delta\mathbf{x}_k = \frac{1}{2}(1 - \mathbf{x}_k^T\mathbf{x}_k) \quad (4.5)$$

But note that $\Delta\mathbf{x}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ and so (4.5) becomes

$$\mathbf{x}_k^T(\mathbf{x}_{k+1} - \mathbf{x}_k) = \frac{1}{2}(1 - \mathbf{x}_k^T\mathbf{x}_k)$$
$$\mathbf{x}_k^T\mathbf{x}_{k+1} = \frac{1}{2}(1 + \mathbf{x}_k^T\mathbf{x}_k) \quad (4.6)$$

Similarly the last component of (4.1) becomes

$$\mathbf{y}_k^T\mathbf{y}_{k+1} = \frac{1}{2}(1 + \mathbf{y}_k^T\mathbf{y}_k) \quad (4.7)$$

From (4.3) and (4.4) we obtain the equations

$$\mathbf{x}_{k+1} = (A_1 - \lambda_k B_1 - \mu_k C_1)^{-1}B_1\mathbf{x}_k\Delta\lambda_k + (A_1 - \lambda_k B_1 - \mu_k C_1)^{-1}C_1\mathbf{x}_k\Delta\mu_k \quad (4.8)$$
$$\mathbf{y}_{k+1} = (A_2 - \lambda_k B_2 - \mu_k C_2)^{-1}B_2\mathbf{y}_k\Delta\lambda_k + (A_2 - \lambda_k B_2 - \mu_k C_2)^{-1}C_2\mathbf{y}_k\Delta\mu_k \quad (4.9)$$

Multiplying (4.8) on the left by $\mathbf{x}_k$ and similarly multiplying (4.9) on the left by $\mathbf{y}_k$ and using (4.6) & (4.7) we obtain the following matrix form equation:

$$\begin{bmatrix} \mathbf{x}_k^T(A_1 - \lambda_k B_1 - \mu_k C_1)^{-1}B_1\mathbf{x}_k & \mathbf{x}_k^T(A_1 - \lambda_k B_1 - \mu_k C_1)^{-1}C_1\mathbf{x}_k \\ \mathbf{y}_k^T(A_2 - \lambda_k B_2 - \mu_k C_2)^{-1}B_2\mathbf{y}_k & \mathbf{y}_k^T(A_2 - \lambda_k B_2 - \mu_k C_2)^{-1}C_2\mathbf{y}_k \end{bmatrix} \begin{bmatrix} \Delta\lambda_k \\ \Delta\mu_k \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(1 + \mathbf{x}_k^T\mathbf{x}_k) \\ \frac{1}{2}(1 + \mathbf{y}_k^T\mathbf{y}_k) \end{bmatrix} \quad (4.10)$$

Now the large cost in solving this system comes from calculating the vectors $(A_i - \lambda_k B_i - \mu_k C_i)^{-1}B_i\mathbf{x}_k$ and $(A_i - \lambda_k B_i - \mu_k C_i)^{-1}B_i\mathbf{y}_k$ for $i = 1, 2$. Each one of these $n \times n$ dimensional systems requires $O(n^3)$ operations to solve them, and so overall the cost of solving these is $O(4n^3)$. If we compare this with the cost of solving (4.1) directly we see that we have halved the number of operations required. The full algorithm for computing Newton's method for a two parameter eigenvalue problem is shown in the next section.

4.2 Practical Newton's method algorithm

The method for solving a two parameter eigenvalue problem by using Newton's method is as follows, and closely followed that given in [1]:

Start with initial approximations $\mathbf{x}_0, \mathbf{y}_0, \lambda_0, \mu_0$.
For $k \in \mathbb{Z}^+$

1. Perform solves of the following systems to find $\mathbf{v}_k, \mathbf{w}_k, \mathbf{p}_k, \mathbf{q}_k$:

$$\begin{aligned} [A_1 - \lambda_k B_1 - \mu_k C_1]\,\mathbf{v}_k &= B_1 \mathbf{x}_k \\ [A_1 - \lambda_k B_1 - \mu_k C_1]\,\mathbf{w}_k &= C_1 \mathbf{x}_k \\ [A_2 - \lambda_k B_2 - \mu_k C_2]\,\mathbf{p}_k &= B_1 \mathbf{y}_k \\ [A_2 - \lambda_k B_2 - \mu_k C_2]\,\mathbf{q}_k &= C_1 \mathbf{y}_k \end{aligned}$$

2. Set up and solve the following system (derived from (4.10))

$$\begin{bmatrix} \mathbf{x}_k^T \mathbf{v}_k & \mathbf{x}_k^T \mathbf{w}_k \\ \mathbf{y}_k^T \mathbf{p}_k & \mathbf{y}_k^T \mathbf{q}_k \end{bmatrix} \begin{bmatrix} \Delta\lambda_k \\ \Delta\mu_k \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(\mathbf{x}_k^T \mathbf{x}_k + 1) \\ \frac{1}{2}(\mathbf{y}_k^T \mathbf{y}_k + 1) \end{bmatrix}$$

3. Solve the following systems to find the updated approximate to the eigenvectors

$$\begin{aligned} [A_1 - \lambda_k B_1 - \mu_k C_1]\,\mathbf{x}_{k+1} &= \Delta\lambda_k B_1 \mathbf{x}_k + \Delta\mu_k C_1 \mathbf{x}_k \\ [A_2 - \lambda_k B_2 - \mu_k C_2]\,\mathbf{x}_{k+1} &= \Delta\lambda_k B_2 \mathbf{x}_k + \Delta\mu_k C_2 \mathbf{x}_k \end{aligned}$$

4. Update the approximations to the eigenvalues in the following way

$$\begin{aligned} \lambda_{k+1} &= \lambda_k + \Delta\lambda_k \\ \mu_{k+1} &= \mu_k + \Delta\mu_k \end{aligned}$$

The Matlab code that implements this algorithm can be found in Appendix A.

4.3 Generalising the diagonal case

To test Newton's method for solving a two parameter eigenvalue problem, the model example containing just diagonal matrices is not sufficient. We can, however, modify the system found in §2.3 to give a less trivial system with much less obvious eigenpairs. We do this by multiplying A_i, B_i and C_i on the left by a nonsingular matrix X and on the right by a nonsingular matrix Y. The eigenvalues remain the same but the eigenvectors become $\mathbf{x} \mapsto Y^{-1}\mathbf{x}$, as

$$A_i \mathbf{x} = \lambda B_i \mathbf{x} + \mu C_i \mathbf{x}$$

Multiplying on the left by X gives

$$X A_i \mathbf{x} = \lambda X B_i \mathbf{x} + \mu X C_i \mathbf{x}$$

which is equivalent to

$$XA_iY(Y^{-1}\mathbf{x}) = \lambda XB_iY(Y^{-1}\mathbf{x}) + \mu XC_iY(Y^{-1}\mathbf{x})$$

So if we take the model example constructed in §2.3 and consider the matrices $X = \begin{bmatrix} 11 & 8 \\ 12 & -1 \end{bmatrix}$ and $Y = \begin{bmatrix} 4 & 25 \\ 0.6 & 13 \end{bmatrix}$ then we obtain the following system:

$$\begin{bmatrix} 53.6 & 483 \\ 46.8 & 274 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} 151.2 & 1241 \\ 141.6 & 848 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \mu \begin{bmatrix} 248.8 & 1999 \\ 236.4 & 1422 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{4.11}$$

$$\begin{bmatrix} 92.8 & 654 \\ 95.4 & 587 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \lambda \begin{bmatrix} 395.2 & 3136 \\ 378.6 & 2283 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + \mu \begin{bmatrix} 492.8 & 3894 \\ 473.4 & 2857 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \tag{4.12}$$

Now from the discussion above, and the analytic calculations made in §2.3 we see that for the system (4.11) and (4.12) we have the following eigenpairs:

$$\mathbf{x} = \frac{1}{\sqrt{169.36}} \begin{bmatrix} 13 \\ -0.6 \end{bmatrix}, \mathbf{y} = \frac{1}{\sqrt{169.36}} \begin{bmatrix} 13 \\ -0.6 \end{bmatrix}, (\lambda, \mu) = (0, 1/5) \tag{4.13}$$

$$\mathbf{x} = \frac{1}{\sqrt{169.36}} \begin{bmatrix} 13 \\ -0.6 \end{bmatrix}, \mathbf{y} = \frac{1}{\sqrt{641}} \begin{bmatrix} -25 \\ 4 \end{bmatrix}, (\lambda, \mu) = (-1/2, 1/2) \tag{4.14}$$

$$\mathbf{x} = \frac{1}{\sqrt{641}} \begin{bmatrix} -25 \\ 4 \end{bmatrix}, \mathbf{y} = \frac{1}{\sqrt{169.36}} \begin{bmatrix} 13 \\ -0.6 \end{bmatrix}, (\lambda, \mu) = (-8/5, 7/5) \tag{4.15}$$

$$\mathbf{x} = \frac{1}{\sqrt{641}} \begin{bmatrix} -25 \\ 4 \end{bmatrix}, \mathbf{y} = \frac{1}{\sqrt{641}} \begin{bmatrix} -25 \\ 4 \end{bmatrix}, (\lambda, \mu) = (-1, 1) \tag{4.16}$$

In §4.4 we will use the knowledge of these exact eigenpairs to numerically calculate errors and hence confirm quadratic convergence of Newton's method.

4.4 Convergence of Newton's method

In this section we consider Newton's method applied to the two parameter eigenvalue problem given by equations (4.11) & (4.12) in §4.3

The tables below show the error in the 2-norm of the approximated eigenvectors generated by Newton's method to that of the exact eigenvector computed analytically.

Table 4.1 shows the results of Newton's method implemented with the following starting guesses:

$$\mathbf{x}_0 = \begin{bmatrix} 1 \\ -0.05 \end{bmatrix}, \mathbf{y}_0 = \begin{bmatrix} 1 \\ -0.05 \end{bmatrix},$$
$$\lambda_0 = \rho_1 \approx -1.99433 \times 10^{-2}, \mu_0 = \rho_2 \approx 1.85766 \times 10^{-1}$$

Where the ρ_i are the components of the tensor Rayleigh quotient that will be discussed later in §5.2.

Step number	Error in $\mathbf{x}$	Error in λ
1	3.39457×10^{-03}	2.36023×10^{-02}
2	6.85614×10^{-04}	3.53853×10^{-03}
3	4.96558×10^{-06}	1.20493×10^{-04}
4	1.35104×10^{-10}	1.96249×10^{-08}
5	2.87896×10^{-14}	5.12581×10^{-16}

Table 4.1: Newton's method converging quadratically to an eigenvalue

In this case, as our starting guess was 'near' the eigenpair (4.13), Newton's method converges to this eigenpair. From this table, we can see that the error in the eigenvalue λ at each subsequent iteration is approximately the square of the previous error. This is indicative of the quadratic convergence of Newton's method.

Table 4.2 shows the convergence obtained when Newton's method is applied to the two parameter eigenvalue problem (4.11) & (4.12) with the following starting guesses:

$$\mathbf{x}_0 = \begin{bmatrix} -0.5 \\ 0.05 \end{bmatrix}, \mathbf{y}_0 = \begin{bmatrix} -1 \\ 0.2 \end{bmatrix}, \lambda_0 = -3, \mu_0 = 3$$

Step number	Error in $\mathbf{x}$	Error in λ
1	7.43806×10^{-01}	8.20811×10^{-00}
2	2.22008×10^{-01}	5.10458×10^{-00}
3	3.44538×10^{-01}	8.63970×10^{-01}
4	4.00360×10^{-02}	3.77521×10^{-01}
5	1.08919×10^{-02}	1.06766×10^{-01}
6	1.04757×10^{-03}	1.02231×10^{-02}
7	9.73147×10^{-06}	9.51748×10^{-05}
8	8.41987×10^{-10}	8.23472×10^{-10}
9	1.56556×10^{-15}	5.81500×10^{-17}

Table 4.2: Newton convergence to a different eigenvalue

With these starting guesses, Newton's method converges to (4.14). This is again characteristic of Newton's method in the fact that it does not necessarily converge to the same eigenpair given different starting points. From *Table 4.2* we can see that convergence is slower to start but increases as the approximations get closer to the eigenpair.

4.5 Three point problem

In this section we shall apply Newton's method to solve a real world problem. The equation

$$\frac{d^2y}{dx^2} + (\lambda + \mu\cos x)y = 0 \tag{4.17}$$

along with the boundary conditions

$$y(0) = y(2.5) = y(5) = 0 \tag{4.18}$$

models a 1-dimensional rod that is fixed at its two ends and its midpoint.

Discretising this ODE using standard finite differences[1] produces the following two parameter eigenvalue problem:

For $n \in 2, 3, 4, \ldots$ Take $h = \frac{1}{n-1}$ and set $x_{1i} = ih$ and $x_{2i} = ih + 2.5$ for $i \in 1, \ldots, n$.

$$A_1 = A_2 = \frac{1}{h^2} tridiag(1, -2, 1) \in \mathbb{R}^{n\times n} \tag{4.19}$$

$$B_1 = B_2 = I_n \in \mathbb{R}^{n\times n} \tag{4.20}$$

$$C_1 = diag(\cos(x_{11}), \ldots, \cos(x_{1n})) \in \mathbb{R}^{n\times n} \tag{4.21}$$

$$C_2 = diag(\cos(x_{21}), \ldots, \cos(x_{2n})) \in \mathbb{R}^{n\times n} \tag{4.22}$$

Table 4.3 shows the convergence obtained when applying Newton's method to the above system in the case when $n = 100$ and taking the initial starting approximations

$$\mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \mathbf{y}_0 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \lambda_0 = -6, \mu_0 = 1$$

These starting conditions converge to the eigenvalue

$$(\lambda, \mu) = (-9.4818 \times 10^0, 4.3577 \times 10^{-14})$$

We can see that, in this case, our starting guess was not sufficiently close to the eigenpair for quadratic convergence to be observed from the first iteration, however once the change in λ gets to $O(10^{-02})$ we then see quadratic convergence. There is an anomaly that occurs in the final step, as this seems far from quadratic convergence. This may be partially explained by the fact that we are not taking the true error from the exact eigenvalue but instead just looking at the change in λ_k, but further investigation into this may be necessary.

Step number	$\|\|\mathbf{x}_k - \mathbf{x}_{k-1}\|\|_2$	$\|\lambda_k - \lambda_{k-1}\|$
1	5.45277×10^{00}	2.13510×10^{00}
2	2.68216×10^{00}	6.95424×10^{-01}
3	1.25658×10^{00}	3.65336×10^{-01}
4	4.91617×10^{-01}	1.98362×10^{-01}
5	1.08447×10^{-01}	7.90685×10^{-02}
6	5.84610×10^{-03}	8.49670×10^{-03}
7	1.70881×10^{-05}	4.99294×10^{-05}
8	1.46004×10^{-10}	8.64489×10^{-10}
9	1.28544×10^{-14}	1.37565×10^{-12}

Table 4.3: Newton convergence of a three point problem

Chapter 5

The Rayleigh quotient method

Another numerical technique that can be used to solve two parameter eigenvalue problems is the Rayleigh quotient method. In this chapter we shall describe the algorithm that does so, and show that whilst it is derived from the Kronecker product form, the dimension of the problem does not rise.

5.1 The basic Rayleigh quotient method

A generalised eigenvalue problem of the form

$$A\mathbf{x} = \lambda B\mathbf{x} \tag{5.1}$$

can be solved using Rayleigh quotient iteration[5]. Firstly we must define the relevant Rayleigh quotient.

Definition 5.1.1. *For given matrices A & B and a vector $\boldsymbol{x}$, we define*

$$\rho(\boldsymbol{x}) := \frac{\boldsymbol{x}^T A\boldsymbol{x}}{\boldsymbol{x}^T B\boldsymbol{x}} \tag{5.2}$$

to be the Rayleigh quotient of the vector $\boldsymbol{x}$.

With this definition in hand, we can define the Rayleigh quotient method for solving equations of the form (5.1).

Form a starting guess $\mathbf{x}_0$ with $||\mathbf{x}_0|| = 1$ as an approximation to the eigenvalue $\mathbf{x}$, then, for $k \in \mathbb{Z}^+$

1. Calculate the Rayleigh quotient:

$$\rho_k = \frac{\mathbf{x}_k^T A\mathbf{x}_k}{\mathbf{x}_k^T B\mathbf{x}_k} \tag{5.3}$$

2. Solve the system
$$(A - \rho_k B)\mathbf{x}_{k+1} = B\mathbf{x}_k \tag{5.4}$$

3. Normalise $\mathbf{x}_{k+1}$:
$$\mathbf{x}_{k+1} = \frac{\mathbf{x}_{k+1}}{||\mathbf{x}_{k+1}||} \tag{5.5}$$

Now Rayleigh quotient iteration is precisely the same method as inverse iteration with the shift taken to be the Rayleigh quotient. So we have that for an intitial approximation $\mathbf{x}_0$ sufficiently close to the eigenvector $\mathbf{x}$ this method will converge to the eigenvector, and the Rayleigh quotient will converge to the corresponding eigenvalue. To see the latter, consider the Rayleigh quotient of A at its eigenvalue $\mathbf{x}^*$

$$\rho(\mathbf{x}^*) = \frac{\mathbf{x}^{*T} A\mathbf{x}^*}{\mathbf{x}^{*T} B\mathbf{x}^*} = \frac{\mathbf{x}^{*T} \lambda B\mathbf{x}^*}{\mathbf{x}^{*T} B\mathbf{x}^*} = \lambda \frac{\mathbf{x}^{*T} B\mathbf{x}^*}{\mathbf{x}^{*T} B\mathbf{x}^*} = \lambda \tag{5.6}$$

This method of solving an eigenvalue problem can be generalised to give us a technique for solving a two parameter eigenvalue problem , as we will discuss in the next section.

5.2 The tensor Rayleigh quotient

Suppose we have a two parameter eigenvalue problem as in (2.1) and (2.2), and we have some approximations $(\mathbf{x}_k, \mathbf{y}_k)$ to the eigenvectors $(\mathbf{x}, \mathbf{y})$. Then we define the tensor Rayleigh quotient in the following way:

Definition 5.2.1. *The tensor Rayleigh quotient* $\rho(\boldsymbol{x}, \boldsymbol{y}, A_1, B_1, C_1, A_2, B_2, C_2)$ *is an ordered pair* (ρ_1, ρ_2) *such that*

$$\rho_1 = \frac{\boldsymbol{z}^T \Delta_1 \boldsymbol{z}}{\boldsymbol{z}^T \Delta_0 \boldsymbol{z}} \tag{5.7}$$

$$\rho_2 = \frac{\boldsymbol{z}^T \Delta_2 \boldsymbol{z}}{\boldsymbol{z}^T \Delta_0 \boldsymbol{z}} \tag{5.8}$$

Where Δ_0, Δ_1 *and* Δ_2 *are as defined in (2.5) ,(2.6) and (2.7) respectively and* $\boldsymbol{z} = \boldsymbol{x} \otimes \boldsymbol{y}$.

Now suppose we have $\mathbf{x}^*$ and $\mathbf{y}^*$ exact eigenvectors for the system. The equations (2.3) and (2.4) hold from §2, ie

$$\Delta_1 \mathbf{z} = \lambda \Delta_0 \mathbf{z}$$

and

$$\Delta_2 \mathbf{z} = \mu \Delta_0 \mathbf{z}$$

where

$$\mathbf{z} = \mathbf{x}^* \otimes \mathbf{y}^*$$

Hence the tensor Rayleigh quotient at an exact eigenvector is

$$\rho_1 = \frac{\mathbf{z}^T \Delta_1 \mathbf{z}}{\mathbf{z}^T \Delta_0 \mathbf{z}} = \frac{\lambda \mathbf{z}^T \Delta_0 \mathbf{z}}{\mathbf{z}^T \Delta_0 \mathbf{z}} = \lambda \tag{5.9}$$

$$\rho_2 = \frac{\mathbf{z}^T \Delta_2 \mathbf{z}}{\mathbf{z}^T \Delta_0 \mathbf{z}} = \frac{\mu \mathbf{z}^T \Delta_0 \mathbf{z}}{\mathbf{z}^T \Delta_0 \mathbf{z}} = \mu \tag{5.10}$$

Now if we compare this with the Rayleigh quotient for the basic eigenvalue problem in (5.6) we can see that they both give the eigenvalue when evaluated at an eigenvector.

Now questions arise about the efficiency of this method, as at each step it appears we must work with the much larger matrices Δ_i. However, we can make use of the mixed product property of the Kronecker product (2.1.1) to show that this is not the case.

Consider $\mathbf{z}^T \Delta_0 \mathbf{z}$. Then we have

$$\begin{aligned} \mathbf{z}^T \Delta_0 \mathbf{z} &= \mathbf{z}^T (B_1 \otimes C_2 - C_1 \otimes B_2)\mathbf{z} \\ &= (\mathbf{x}^T \otimes \mathbf{y}^T)(B_1 \otimes C_2 - C_1 \otimes B_2)(\mathbf{x} \otimes \mathbf{y}) \\ &= (\mathbf{x}^T \otimes \mathbf{y}^T)(B_1\mathbf{x} \otimes C_2\mathbf{y} - C_1\mathbf{x} \otimes B_2\mathbf{y}) \\ &= \mathbf{x}^T B_1\mathbf{x} \otimes \mathbf{y}^T C_2\mathbf{y} - \mathbf{x}^T C_1\mathbf{x} \otimes \mathbf{y}^T B_2\mathbf{y} \end{aligned} \tag{5.11}$$

But note that $\mathbf{x}^T B_1\mathbf{x}$, $\mathbf{y}^T C_2\mathbf{y}$ etc. are scalars. Hence $\otimes$ becomes normal multiplication. So (5.11) becomes

$$\mathbf{z}^T \Delta_0 \mathbf{z} = (\mathbf{x}^T B_1\mathbf{x})(\mathbf{y}^T C_2\mathbf{y}) - (\mathbf{x}^T C_1\mathbf{x})(\mathbf{y}^T B_2\mathbf{y}) \tag{5.12}$$

Now consider $\mathbf{z}^T \Delta_2 \mathbf{z}$. By the same arguments as before we obtain

$$\begin{aligned} \mathbf{z}^T \Delta_2 \mathbf{z} &= \mathbf{z}^T (B_1 \otimes A_2 - A_1 \otimes B_2)\mathbf{z} \\ &= (\mathbf{x}^T \otimes \mathbf{y}^T)(B_1 \otimes A_2 - A_1 \otimes B_2)(\mathbf{x} \otimes \mathbf{y}) \\ &= (\mathbf{x}^T \otimes \mathbf{y}^T)(B_1\mathbf{x} \otimes A_2\mathbf{y} - A_1\mathbf{x} \otimes B_2\mathbf{y}) \\ &= \mathbf{x}^T B_1\mathbf{x} \otimes \mathbf{y}^T A_2\mathbf{y} - \mathbf{x}^T A_1\mathbf{x} \otimes \mathbf{y}^T B_2\mathbf{y} \\ &= (\mathbf{x}^T B_1\mathbf{x})(\mathbf{y}^T A_2\mathbf{y}) - (\mathbf{x}^T A_1\mathbf{x})(\mathbf{y}^T B_2\mathbf{y}) \end{aligned} \tag{5.13}$$

Hence by (5.12) and (5.13) we get

$$\rho_2 = \frac{(\mathbf{x}^T B_1\mathbf{x})(\mathbf{y}^T A_2\mathbf{y}) - (\mathbf{x}^T A_1\mathbf{x})(\mathbf{y}^T B_2\mathbf{y})}{(\mathbf{x}^T B_1\mathbf{x})(\mathbf{y}^T C_2\mathbf{y}) - (\mathbf{x}^T C_1\mathbf{x})(\mathbf{y}^T B_2\mathbf{y})} \tag{5.14}$$

Similarly we find that

$$\rho_1 = \frac{(\mathbf{x}^T C_1\mathbf{x})(\mathbf{y}^T A_2\mathbf{y}) - (\mathbf{x}^T A_1\mathbf{x})(\mathbf{y}^T C_2\mathbf{y})}{(\mathbf{x}^T B_1\mathbf{x})(\mathbf{y}^T C_2\mathbf{y}) - (\mathbf{x}^T C_1\mathbf{x})(\mathbf{y}^T B_2\mathbf{y})} \tag{5.15}$$

These last two equations (5.14) and (5.15) show that we do not need to use the Kronecker product in calculating the tensor Rayleigh quotient.

5.3 Tensor Rayleigh Quotient Iteration

The following is an algorithm that describes Tensor Rayleigh Quotient Iteration (TRQI). In essence it is just Newton's method where instead of taking the approximations to the eigenvalues at each step to be (λ_k, μ_k) we take the tensor Rayleigh quotient $(\rho_{k,1}, \rho_{k,2})$.

Start with initial approximations to the eigenvectors, $\mathbf{x}_0$ and $\mathbf{y}_0$. Then for $k \in \mathbb{Z}^+$:

1. Calculate the tensor Rayleigh quotient

$$(\rho_{k,1}, \rho_{k,2}) = \rho(\mathbf{x}_k, \mathbf{y}_k, A_1, B_1, C_1, A_2, B_2, C_2) \tag{5.16}$$

2. Check the determinant of the matrices $[A_1 - \rho_{k,1}B_1 - \rho_{k,2}C_1]$ and $[A_2 - \rho_{k,1}B_2 - \rho_{k,2}C_2]$ and if either are equal to 0, perturb $(\rho_{k,1}, \rho_{k,2})$ slightly.

3. Solve the following equations for $\mathbf{v}_k, \mathbf{w}_k, \mathbf{p}_k$ and $\mathbf{q}_k$:

$$\begin{aligned}
[A_1 - \rho_{k,1}B_1 - \rho_{k,2}C_1]\,\mathbf{v}_k &= B_1\mathbf{x}_k \\
[A_1 - \rho_{k,1}B_1 - \rho_{k,2}C_1]\,\mathbf{w}_k &= C_1\mathbf{x}_k \\
[A_2 - \rho_{k,1}B_2 - \rho_{k,2}C_2]\,\mathbf{p}_k &= B_1\mathbf{y}_k \\
[A_2 - \rho_{k,1}B_2 - \rho_{k,2}C_2]\,\mathbf{q}_k &= C_1\mathbf{y}_k
\end{aligned}$$

4. Set up and solve the following system

$$\begin{bmatrix} \mathbf{x}_k^T\mathbf{v}_k & \mathbf{x}_k^T\mathbf{w}_k \\ \mathbf{y}_k^T\mathbf{p}_k & \mathbf{y}_k^T\mathbf{q}_k \end{bmatrix} \begin{bmatrix} \Delta\lambda_k \\ \Delta\mu_k \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(\mathbf{x}_k^T\mathbf{x}_k + 1) \\ \frac{1}{2}(\mathbf{y}_k^T\mathbf{y}_k + 1) \end{bmatrix}$$

5. Update the approximate eigenvectors

$$\begin{aligned}
\mathbf{x}_{k+1} &= \Delta\lambda_k\mathbf{v}_k + \Delta\mu_k\mathbf{w}_k \\
\mathbf{y}_{k+1} &= \Delta\lambda_k\mathbf{p}_k + \Delta\mu_k\mathbf{q}_k
\end{aligned}$$

6. Normalise the approximated eigenvectors

$$\begin{aligned}
\mathbf{x}_{k+1} &= \frac{\mathbf{x}_{k+1}}{||\mathbf{x}_{k+1}||} \\
\mathbf{y}_{k+1} &= \frac{\mathbf{y}_{k+1}}{||\mathbf{y}_{k+1}||}
\end{aligned}$$

Repeat these steps until, for a given small value of ϵ,

$$\sqrt{||(A_1 - \rho_{k,1}B_1 - \rho_{k,2}C_1)\mathbf{x}_{k+1}||^2 + ||(A_2 - \rho_{k,1}B_2 - \rho_{k,2}C_2)\mathbf{y}_{k+1}||^2} < \epsilon \tag{5.17}$$

More information on the choice of this termination condition can be found in [6], but it is clear that at the root, the computed value will be 0.

5.4 Numerics for TRQI

A program was written in Matlab to implement TRQI, the code for which can be found in Appendices B and C. In this section we shall look at the rates of convergence of this algorithm, and compare it with that of Newton's method.

5.4.1 The generalised diagonal example

Let us take as our two parameter eigenvalue problem the same problem as constructed in §4.3. That is to say, we wish to solve

$$\begin{bmatrix} 53.6 & 483 \\ 46.8 & 274 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} 151.2 & 1241 \\ 141.6 & 848 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \mu \begin{bmatrix} 248.8 & 1999 \\ 236.4 & 1422 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{5.18}$$

$$\begin{bmatrix} 92.8 & 654 \\ 95.4 & 587 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \lambda \begin{bmatrix} 395.2 & 3136 \\ 378.6 & 2283 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + \mu \begin{bmatrix} 492.8 & 3894 \\ 473.4 & 2857 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \tag{5.19}$$

Now in order to have a comparison with Newton's method, we look at initial approximations to the eigenvectors that will converge to the same eigenpair as Newton's method converged to in §4.4. Thus we have

$$\begin{aligned} \mathbf{x}_0 &= \begin{bmatrix} 1 \\ -0.05 \end{bmatrix} \\ \mathbf{y}_0 &= \begin{bmatrix} 1 \\ -0.05 \end{bmatrix} \end{aligned}$$

Note again that using TRQI we need not have a starting guess for our eigenvalues. Table 5.1 shows the convergence obtained when applying TRQI to (5.18) and (5.19) with the above initial approximations.

Step number	Error in $\mathbf{x}$	Error in λ
1	7.75736×10^{-03}	3.45707×10^{-02}
2	3.97636×10^{-03}	6.88606×10^{-03}
3	3.09372×10^{-04}	1.50243×10^{-03}
4	4.76090×10^{-06}	7.69562×10^{-08}
5	9.24207×10^{-11}	5.79702×10^{-10}
6	2.30660×10^{-16}	2.06656×10^{-16}

Table 5.1: TRQI convergence to the eigenvalue (4.13)

From this table we can see that the rate of convergence is not constant. In a similar fashion to the convergence detailed in Table 4.3 we see that the convergence is quadratic after a few iterations, and then proceeds to slow down as the eigenpair is almost attained. We can directly compare these results with that obtained in §4.4, as the problem and the starting values are the same in

this case and in *Table 4.1.* From this we can conjecture that Newton's method is more efficient for solving a two parameter eigenvalue problem than TRQI. There is more evidence to support this as in the course of testing TRQI on different systems it can be seen that often the method will not converge at all, when given the same initial approximate eigenvectors and arbitrary initial eigenvalues Newton's method will find an eigenpair.

There are some interesting questions that have emerged as a result of writing this thesis. It might be the case that the algorithm stated in §5.3 can be derived from the standard Rayleigh quotient method applied to the system (2.3) and (2.4), however time constraints meant we were unable to investigate this fully in this paper.

Finally, in the course of researching this thesis, an error in [6] was found that should be noted for further reference to help when implementing TRQI. (5.14) is the correct formula for ρ_2 as opposed to the formula given in §4 of [6].

Bibliography

[1] Hochstenbach, M.E., Košir, T. & Plestenjak, B., 2002. A Jacobi-Davidson type method for the two-parameter eigenvalue problem. SIAM Journal on Scientific Computing.

[2] Plestenjak, B., 2003. Lecture slides, Numerical methods for algebraic two-parameter eigenvalue problems. Ljubljana, University of Ljubljana.

[3] Horn, R. A., 1994. Topics in Matrix Analysis. Cambridge, Cambridge University Press

[4] Spence, A. & Graham, I.G., 2002. Numerical Methods for Bifurcation Problems. Bath, University of Bath.

[5] Chatelin, F., 1993. Eigenvalues of Matrices. Chichester, John Wiley & Sons Ltd.

[6] Plestenjak, B., 2000. A continuation method for a right definite two-parameter eigenvalue problem. Ljubljana, University of Ljubljana.

Further Reading

[7] Plestenjak, B., 2003. Numerical methods for algebraic two-parameter eigenvalue problems. Ljubljana, University of Ljubljana. Banff.

[8] Atkinson, F. V., 1972. Multiparameter Eigenvalue Problems. New York, Academic Press.

Appendix A

Matlab code for implementing Newton's method

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%A Program to implement Newton's method
%to find approxiate solutions for the
%Two Parameter Eigenvalue Problem
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x,y,lambda,mu]
  = Newton(A1,B1,C1,A2,B2,C2,x,y,lambda,mu)

clc
format long

test = 1;

while test > 10^-10

%denote (A_1 - lambda_k B_1 - mu_k C_1)  by k1inv
%denote (A_2 - lambda_k B_2 - mu_k C_2)  by k2inv

k1inv = (A1 - lambda * B1 - mu * C1);
k2inv = (A2 - lambda * B2 - mu * C2);

%calculate the v_ks, w_ks, p_ks and q_ks.
v = k1inv \ (B1*x);
w = k1inv \ (C1*x);
p = k2inv \ (B2*y);
```

```
q = k2inv \ (C2*y);


%construct 2 x 2 matrix by premultiplying by x_k' and y_k'
xktvk = x' * v;
xktwk = x' * w;
yktpk = y' * p;
yktqk = y' * q;
%now set into the matrix:
left = [xktvk xktwk; yktpk yktqk];
right = 0.5* [(x'*x + 1);(y'*y +1)];

%calculate the deltalambda_k and deltamu_k
Delta = left \ right;

Deltalambda = Delta(1,1)
Deltamu     = Delta(2,1);

%construct (4.3) and (4.3) of Plestenjak
xupdater = Deltalambda * B1 * x + Deltamu * C1 * x;
yupdater = Deltalambda * B2 * y + Deltamu * C2 * y;

difference = x;
%solve these and hence update x and y
x = k1inv \ xupdater;
y = k2inv \ yupdater;


%update lambda and mu approximations
lambda = lambda + Deltalambda;
mu     = mu     + Deltamu;

difference = x - difference;
test = norm(difference)
end
```

Appendix B

Matlab code for calculating the Rayleigh quotient

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% A function file to calculate the Tensor Rayleigh   %%
%%% quotient of a two parameter eigenvalue system    %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [rho1,rho2] = tensorrayleigh(x,y,A1,B1,C1,A2,B2,C2)

denom = (x'*B1*x)*(y'*C2*y) - (x'*C1*x)*(y'*B2*y);

rho1num = (x'*C1*x)*(y'*A2*y) - (x'*A1*x)*(y'*C2*y);
rho2num = (x'*B1*x)*(y'*A2*y) - (x'*A1*x)*(y'*B2*y);

rho1 = rho1num/denom;
rho2 = rho2num/denom;
```

Appendix C

Matlab code for implementing the Rayleigh quotient method

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%A Program to implement the tensor Rayleigh quotient method
%to find approxiate solutions for the
%Two Parameter Eigenvalue Problem
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x,y,rho1,rho2,convergence]
= trayleigh(A1,B1,C1,A2,B2,C2,x,y)

clc
format long


%set the accuracy that you require in ep2
ep2 = 10^-07;




terminate = 1;
convergence = [0 0 0 0];

while terminate > ep2

%step 1 - calculate the Tensor Rayleigh quotient:
[rho1,rho2] = tensorrayleigh(x,y,A1,B1,C1,A2,B2,C2);
```

```
%the following line to produces a matrix detailing
%the steps taken until convergence:
convergence = [convergence; [x(1) y(1) rho1 rho2]];

%denote (A_1 - rho1_k B_1 - rho2_k C_1)  by k1inv
%denote (A_2 - rho1_k B_2 - rho2_k C_2)  by k2inv

k1inv = (A1 - rho1 * B1 - rho2 * C1);
k2inv = (A2 - rho1 * B2 - rho2 * C2);



%check for singularity and perturb if necessary:
if det(k1inv) == 0 | det(k2inv) == 0
    rho1 = rho1 + 10^-6;
    rho2 = rho2 + 10^-6;
    k1inv = (A1 - rho1 * B1 - rho2 * C1);
    k2inv = (A2 - rho1 * B2 - rho2 * C2);
else
end



%calculate the v_ks, w_ks, p_ks and q_ks.
v = k1inv \ (B1*x);
w = k1inv \ (C1*x);
p = k2inv \ (B2*y);
q = k2inv \ (C2*y);


%construct 2 x 2 matrix by premultiplying by x_k' and y_k'
xktvk = x' * v;
xktwk = x' * w;
yktpk = y' * p;
yktqk = y' * q;
%now set into the matrix:
left = [xktvk xktwk; yktpk yktqk];
right = 0.5* [(x'*x + 1);(y'*y +1)];

%calculate the deltalambda_k and deltamu_k
Delta = left \ right;

Deltalambda = Delta(1,1);
Deltamu     = Delta(2,1);
```

```
%calculate the new values for x and y:
xupdater = Deltalambda * v + Deltamu * w;
yupdater = Deltalambda * p + Deltamu * q;

x = xupdater / norm(xupdater);
y = yupdater / norm(yupdater);


%calculate the value that will decide if the function should
%terminate and return the eigenpairs.

terminate = sqrt((norm(k1inv*x))^2 + (norm(k2inv*y))^2);
end
```