Application of Control Techniques to Solving Linear Systems of Equations

Mofozul Ali

September 1994

Submitted to :

The University of Reading

Department of Mathematics

in partial fulfilment of the requirements for the

Degree of Master of Science

Abstract

Standard iteration techniques for solving linear systems are discussed. We then analyse what effect the eigenvalues and the eigenvectors of the iteration matrix have on the convergence of the iteration process. Finally, we propose an application of Control Theory to determine the eigenvalues and the eigenvectors of the iteration matrix in order to ensure rapid convergence.

Acknowledgements

I would like to thank my supervisor, Prof. M. J. Baines, for his help and guidance during the writing of this dissertation. I would also like to thank Dr. N. K. Nichols for her assistance.

Contents

1 Introduction **3**

2 Iterative Techniques for Solving Linear Systems **5**

2.1 Jacobi iterative method 6

2.2 Gauss-Siedel iterative method 9

2.3 Preconditioned Conjugate Gradient method 10

2.4 General Iteration Method 13

3 Application of Control Techniques in Iterative Methods **24**

3.1 Robust pole assignment 25

3.2 Numerical Algorithm 29

3.3 Applications 31

4 Conclusions **38**

Chapter 1

Introduction

In this dissertation we study numerical methods for solving linear systems of the form :

$$A\mathbf{x} = \mathbf{b} \tag{1.1}$$

where A is a given real $n \times n$ matrix and b is a given real column vector of order n.

For large sparse systems iterative techniques are very efficient in terms of computer storage and time requirements. Systems of this type arise in the numerical solution of boundary-value problems and partial-differential equations.

We discuss standard iteration procedures and analyse the effect on convergence of assigning :

a) eigenvalues

b) eigenvectors

to the iteration matrix.

The analysis, then enables us to use an application of Control Theory to assign

eigenvalues and eigenvectors to the iteration matrix, and achieve our objective, which in the study of iteration techniques is rapid convergence.

Chapter 2

Iterative Techniques for Solving Linear Systems

In this chapter we are going to study numerical methods for solving linear systems of the form :

$$A\mathbf{x} = \mathbf{b} \tag{2.1}$$

where A is a given real $n \times n$ matrix and b is a given real column vector of order n. The solution vector $\mathbf{x}$ exists and is unique if and only if A is nonsingular. The solution is given by :

$$\mathbf{x} = A^{-1}\mathbf{b}. \tag{2.2}$$

Equation (2.1) can be solved using **direct methods** such as L-U decomposition or Gaussian elimination. Methods of this type require A to be factorized and are impractical if A is large and sparse. **Iterative methods** are an alternative to direct methods and are ideally suited to inverting large sparse matrices. These methods involve an initial approximation $\mathbf{x}^0$ to the solution $\mathbf{x}$, and generate a

sequence of vectors $\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3, ...$ according to some algorithm, that is intended to converge to the exact solution $\mathbf{x}$. The study of iterative methods, therefore, focuses on how quickly the iterates $\mathbf{x}^k$ converge. Another advantage of this type of algorithm is that the matrix A is not altered during the computation, as opposed to most direct methods. Therefore, although the computation may be long, rounding errors are much reduced.

2.1 Jacobi iterative method

This is perhaps the simplest iterative scheme [e.g. Burden and Faires (1985)]. It is defined for matrices that have nonzero diagonal elements. Consider a 3-by-3 system $A\mathbf{x} = \mathbf{b}$ and re-write it as follows :

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$

$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$$

$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}.$$

If $\mathbf{x}^k$ is an approximation to $\mathbf{x} = A^{-1}\mathbf{b}$, then a natural way to generate a new approximation $\mathbf{x}^{k+1}$ is to compute :

$$x_1^{(k+1)} = (b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)})/a_{11}$$

$$x_2^{(k+1)} = (b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)})/a_{22}$$

$$x_3^{(k+1)} = (b_3 - a_{31}x_1^{(K)} - a_{32}x_2^{(k)})/a_{33}.$$

This defines the Jacobi iteration for the case when $n = 3$. For general n we have

$$x_i^{(k+1)} = [b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}]/a_{ii} \qquad for \quad i = 1, 2, ..., n. \tag{2.3}$$

Alternatively, we can obtain the matrix iteration form by splitting A into its diagonal and off-diagonal parts. If we let D be the diagonal of A, and $-L$ and $-U$ be the strictly lower and upper triangular parts of A, then $A = D - L - U$. Equation (2.1) now becomes :

$$(D - L - U)\mathbf{x} = \mathbf{b}$$

$$D\mathbf{x} = (L + U)\mathbf{x} + \mathbf{x}$$

$$\mathbf{x} = D^{-1}(L + U)\mathbf{x} + D^{-1}\mathbf{b}$$

Therefore, the matrix form of the iteration is :

$$\mathbf{x}^{(k)} = D^{-1}(L + U)\mathbf{x}^{(k-1)} + D^{-1}\mathbf{b}, \qquad k = 1, 2, ... \tag{2.4}$$

Generally, Equation (2.3) is used in computation.

Example 1: Solve the following linear system:

$$\begin{pmatrix} 10 & -1 & 2 & 0 \\ -1 & 11 & -1 & 3 \\ 2 & -1 & 10 & -1 \\ 0 & 3 & -1 & 8 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 6 \\ 25 \\ -11 \\ 15 \end{pmatrix}$$

A simple program was written in Fortran to carry out the Jacobi iteration given by Equation (2.3), with the initial vector $\mathbf{x}^0 = \mathbf{0}$. The results obtained are shown in Table 1.

Table 1: Jacobi iterative method

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$
0	0.00000000	0.00000000	0.00000000	0.00000000
1	0.60000000	2.27272727	−1.10000000	1.87500000
2	1.04727273	1.71590909	−0.80522727	0.88522727
3	0.93263636	2.05330579	−1.04934091	1.13088068
4	1.01519876	1.95369576	−0.96810863	0.97384272
5	0.98899130	2.01141473	−1.01028590	1.02135051
6	1.00319865	1.99224126	−0.99452174	0.99443374
7	0.99812847	2.00230688	−1.00197223	1.00359431
8	1.00062513	1.99867030	−0.99903558	0.99888839
9	0.99967415	2.00044767	−1.00036916	1.00061919
10	1.00011860	1.99976795	−0.99982814	0.99978598

The decision to stop after ten iterations is based on

$$\frac{\left|\left|\mathbf{x}^{(10)} - \mathbf{x}^{(9)}\right|\right|_\infty}{\left|\left|\mathbf{x}^{(10)}\right|\right|_\infty} \quad < \quad 7 \times 10^{-4}$$

The exact solution is, in fact, $\mathbf{x} = (1, 2, -1, 1)^T$.

2.2 Gauss-Siedel iterative method

The Jacobi iteration does not use the most up-to-date information when computing $x_i^{(k+1)}$. For example, $x_1^{(k)}$ is used in the calculation of $x_2^{(k+1)}$ even though $x_1^{(k+1)}$ is known. The Gauss-Siedel iteration overcomes this, and is defined by :

$$x_i^{(k+1)} = [b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}]/a_{ii} \qquad for \quad i = 1, 2, ..., n. \qquad (2.5)$$

The matrix form of the Gauss-Siedel is given by :

$$\mathbf{x}^{(k)} = (D - L)^{-1}U\mathbf{x}^{(k-1)} + (D - L)^{-1}\mathbf{b}, \qquad k = 1, 2, ... \qquad (2.6)$$

For the lower-triangular matrix $D - L$ to be nonsingular, it is necessary and sufficient that $a_{ii} \neq 0$ for each $i = 1, 2, ..., n$.

Example 2: Solve the following linear system using the Gauss-Siedel iteration :

$$\begin{pmatrix} 10 & -1 & 2 & 0 \\ -1 & 11 & -1 & 3 \\ 2 & -1 & 10 & -1 \\ 0 & 3 & -1 & 8 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 6 \\ 25 \\ -11 \\ 15 \end{pmatrix}$$

By making a small alteration to our Fortran program and again taking $\mathbf{x}^0 = \mathbf{0}$, we obtain the following table of results :

Table 2: Gauss-Siedel iterative method

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$
0	0.00000000	0.00000000	0.00000000	0.00000000
1	0.60000000	2.32727273	−0.98727273	0.87886364
2	1.03018182	2.03693802	−1.01445620	0.98434122
3	1.00658504	2.00355502	−1.00252738	0.99835095
4	1.00086098	2.00029825	−1.00030728	0.99984975
5	1.00009128	2.00002134	−1.00003115	0.99998810

The tolerance level for the stopping criteria was again 7×10^{-4}. Note that Jacobi's method in Example 1 required twice as many iterations for the same accuracy. In general, Gauss-Siedel method is superior to Jacobi although there are examples where the converse is true - See Varga (1962).

2.3 Preconditioned Conjugate Gradient method

The **Conjugate Gradient method** is derived by re-writing $A\mathbf{x} = \mathbf{b}$ as an optimization problem. Assume A is symmetric and positive definite, and let

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b}$$

We need to minimise $F(\mathbf{x})$. The minimum $\mathbf{x}^*$ occurs when :

$$-\nabla F(\mathbf{x} = \mathbf{x}^*) = \mathbf{r} = \mathbf{b} - A\mathbf{x}^* = 0$$

To minimise $F(\mathbf{x})$, the Conjugate Gradient method takes a step α in the direction closest to the gradient $\mathbf{r}^k$ but which is orthogonal to all previous directions. The Conjugate Gradient Method has a very slow rate of convergence but this can

be overcome by **pre-conditioning** A. The Preconditioned Conjugate Gradient method uses a preconditioner M to accelerate convergence [Golub and Van Loan].

The Preconditioned Conjugate Gradient Algorithm :

Input $\mathbf{x}_0$, β_0. Set $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, $\mathbf{p}_{-1} = \mathbf{r}_0$.

Input M, the preconditioner, choose $M = D$ for Jacobi preconditioning.

For $k = 0, 1, 2, ...$ until convergence, do

$Mz_k = r_k$

$\beta_k = r_k^T z_k / r_{k-1}^T z_{k-1}$

$p_k = z_k + \beta_k p_{k-1}$

$\alpha_k = r_k^T z_k / p_k^T A p_k$

$x_{k+1} = x_k + \alpha_k p_k$

$r_{k+1} = b - Ax_k \equiv r_k - \alpha_k A p_k$

Example 3: Use Jacobi-Preconditioned Conjugate Gradient method to solve the following linear system :

$$\begin{pmatrix} 10 & -1 & 2 & 0 \\ -1 & 11 & -1 & 3 \\ 2 & -1 & 10 & -1 \\ 0 & 3 & -1 & 8 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 6 \\ 25 \\ -11 \\ 15 \end{pmatrix}$$

The results obtained from a Fortran program are presented, below, in Table 3 :

Table 3: Preconditioned C.G. method

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$
0	0.00000000	0.00000000	0.00000000	0.00000000
1	0.46461746	1.75991463	−0.85179867	1.45192957
2	1.04214099	1.94075677	−0.91679619	1.12828630
3	1.00488813	1.98938063	−1.01181525	1.00690572
4	1.00000000	2.00000000	−1.00000000	1.0000000

The above method converges after only four iterations and attains the precise solution to machine accuracy.

The Preconditioned Conjugate Gradient Method is much more efficient than either of the two methods introduced before it, yet I have discussed the former two in more detail. This is because the those methods have a very similar form to the control problem we shall encounter in Chapter 3. The following two theorems give the conditions for convergence of the three methods,

Theorem :

If A is strictly diagonally dominant, then both the Jacobi and the Gauss-Siedel methods converge for any choice of $\mathbf{x}^0$.

Theorem : Covergence of the Preconditioned Conjugate Gradient Method is guaranteed if A is symmetric, positive definite and M is also positive definite.

A symmetric matrix A is called **positive definite** if $\mathbf{x}^T A\mathbf{x} > 0$ for every $\mathbf{x} \neq \mathbf{0}$.

An $n \times n$ matrix A is said to be **strictly diagonally dominant** if :

$$|a_{ii}| > \sum_{j=1}^{n} |a_{ij}| \qquad (j \neq i)$$

holds for each $i = 1, 2, ..., n$.

The proof of these theorems are omitted but can be found in Young (1971).

2.4 General Iteration Method

If we multiply the equation $A\mathbf{x} = \mathbf{b}$ by a matrix G, then $GA\mathbf{x} = G\mathbf{b}$

and we may write $\mathbf{x} = (I - GA)\mathbf{x} + G\mathbf{b}$

So a general iteration process is :

$$\mathbf{x}^{n+1} = (I - GA)\mathbf{x}^n + G\mathbf{b} \tag{2.7}$$

where G is to be chosen.

It can easily be seen that :

$G = D^{-1}$ gives Jacobi Iteration method

$G = (D - L)^{-1}$ gives Gauss-Siedel Iteration method

The convergence of this method is dependent on the eigenvalues of $(I - GA)$.

Theorem :

The iterative method, Equation (2.7), is convergent if and only if :

$$\rho\,(I - GA) \quad < \quad 1$$

The proof of this theorem is omitted, but can be found in e.g. Young (1971).

Rates of Convergence : (Young (1971)

The *average rate of convergence* is defined by

$$R_n(I - GA) = -(\frac{1}{n})log\,||(I - GA)^n||$$

The *asymptotic rate of convergence* is defined by

$$R(I - GA) = \lim_{n \to \infty} R_n(I - GA) = -log\ \rho\,(I - GA)$$

We shall refer to $R_n(I - GA)$ as the *rate of convergence*, and see that, eventually, it tends to the asymptotic rate.

The convergence rate of the iteration process depends on the spectrum (i.e. the eigenvalues) of $(I - GA)$. Our aim is to choose G such that the eigenvalues of $(I - GA)$ are near the origin, so that we have a better iteration method than the three presented above.

Since the matrices I and A have full rank, there are well known theorems in Control Theory, which state that the pair (I, A) is always **completely controllable**. This means that there always exists a matrix G, which assigns $(I - GA)$ any specified eigenvalues and eigenvectors, i.e. we can find a matrix G such that

$$(I - GA) = X^{-1}\Lambda X \tag{2.8}$$

where $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_n)$ are the eigenvalues to be assigned, and X is any nonsingular matrix representing the eigenvectors.

In Chapter 3, we will discuss this result in more detail and present an algorithm for finding G. But before that we are going to look at how G affects convergence. We have already stated that the rate of covergence tends to the asymptotic rate which is determined by $\rho(I - GA)$. An efficient iteration method will achieve rapid early convergence and so we are going to study how G, which determines the eigenstructure of $(I - GA)$, affects covergence in the early stages of the iteration process.

In order to see what effect the range of eigenvalues, or the condition number of

the eigenvector matrix has on convergence, we re-arrange Equation (2.8) to get :

$$G = (I - X^{-1}\Lambda X)A^{-1} \tag{2.9}$$

Note that the calculation of G requires inversion of A, and if we are able to perform that operation accurately then the solution to our main problem would be easily found from Equation (2.2). But, in our inverse problem, Equation (2.9) gives us an expression to find G, which we can then use in our General Iteration method, given by Equation (2.7), to examine convergence.

Case 1

First of all we study how the spectrum of $(I - GA)$ affects convergence. Since we are able to assign any eigenvectors to the matrix, in this case we choose $X \equiv I$, so that the condition number of X, $\nu(X) = 1$. Equation (2.9) now becomes :

$$G = (I - \Lambda)A^{-1} \tag{2.10}$$

This equation allows us to find G, for given eigenvalues. A program has been written in MATLAB, which possesses internal functions to perform matrix multiplications and inversions, to calculate G and then carry out the general iteration method [Equation (3.7)].

Example 4: If A and $\mathbf{b}$ are as given in Examples $1-3$ and we assign the eigenvalues of $(I-GA)$ to be $-0.1, -0.05, 0.05$ and 0.1, i.e. $\Lambda = diag(-0.1, -0.05, 0.05, 0.1)$,

then the iteration process terminates after four steps. The rate of convergence after each step is given below:

Table 4: $\Lambda = diag(-0.1, -0.05, 0.05, 0.1)$

k	*convergence rate*
1	2.9957
2	2.3026
3	2.3026
4	2.3026

Notice that $-log(0.1) = 2.3026$, so after just two iterations the asymptotic rate, to machine accuracy, is reached. Naturally, we would expect fast convergence since we have chosen the eigenvalues to be close to the origin. If we assign eigenvalues of similar magnitude to different examples, then we get a similar outcome.

Example 5: Assign $\Lambda = diag(-0.080, -0.035, 0.010, 0.055, 0.100)$ to the following linear system :

$$\begin{pmatrix} 4 & 1 & 3 & 4 & 2 \\ 1 & 0 & 0 & 1 & 0 \\ 3 & -2 & 1 & -1 & 2 \\ -1 & 2 & 0 & 4 & -1 \\ 0 & 1 & -1 & -3 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 10 \\ 0 \\ -1 \\ 7 \\ 3 \end{pmatrix}$$

The exact solution is $\mathbf{x} = (-1.625, 2.000, -0.750, 1.625, 5.125)^T$ and this is attained in four iterations. The convergence rates were 2.3026, 2.3026, 2.3026, and 2.3026 at each iteration.

Example 6: Assign $\Lambda = diag(-0.1, -0.08, -0.06, -0.04, -0.02, 0.00, 0.02, 0.04, 0.06, 0.08)$

to the following linear system :

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 1 & 0 & 4 & 0 & 0 & -1 & 0 & 0 \\ 0 & 3 & 0 & 1 & 0 & 0 & -1 & 0 & 2 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 5 & 0 & 0 & -2 \\ -1 & 0 & 0 & 2 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 1 & -3 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & 1 & 0 \\ 0 & 3 & 0 & 0 & 0 & -2 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 4 & 0 & 0 & -2 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{pmatrix} = \begin{pmatrix} 1 \\ 10 \\ 0 \\ -11 \\ 24 \\ 3 \\ -8 \\ 2 \\ 9 \\ 2 \end{pmatrix}$$

The exact solution is :

$\mathbf{x} = (5.8429, -3.6286, 24.6571, -0.8714, -2.7286, -3.1000, 10.8143, 7.9000, 5.7875, 15.4857)^T$

and this is attained in four iterations, again. The convergence rates were 2.7677, 2.3026, 2.3026, and 2.3026 at each iteration.

We are now going to use A and $\mathbf{b}$ as given in Example 3 and assign various sets of eigenvalues to $(I - GA)$ to see how they affect convergence. We know that the asymptotic rate will be equal to the negative log of the largest eigenvalue of $(I - GA)$. Apart from knowing that particular value, we are also interested in the initial convergence rate, the number of iterations it takes to reach the asymptotic rate, and the number of iterations it takes for the process to stop by satisfying the stopping criteria. Table 5 represents a summary of this for different sets of eigenvalues :

Table 5

$\Lambda = diag(\lambda_i)$	*initial rate of convergence*	*asymptotic rate of convergence*	*No. of iter. for asymp. rate*	*total no. of iterations*
$a) -0.10, -0.05, 0.05, 0.10$	2.99573	2.30259	2	4
$b) -0.30, -0.20, -0.10, 0.2$	1.60944	1.20397	2	7
$c) -0.30, -0.20, -0.10, 0.4$	1.60944	0.91623	2	8
$d) -0.30, -0.20, -0.10, 0.6$	1.20397	0.51083	2	13
$e) -0.60, 0.10, 0.20, 0.30$	1.20397	0.51083	2	15
$f) -0.60, -0.30, -0.20, 0.40$	1.20397	0.51083	2	15
$g) -0.60, -0.30, 0.30, 0.50$	1.20397	0.51083	2	15
$h) -0.60, -0.20, 0.30, 0.50$	1.20397	0.51083	2	15
$i) -0.60, -0.20, -0.20, 0.50$	1.20397	0.5103	2	15
$j) -0.80, -0.30, 0.20, 0.70$	0.91629	0.22314	2	34
$k) -0.80, 0.20, 0.50, 0.70$	0.91629	0.22314	2	34
$l) -0.80, 0.50, 0.60, 0.70$	0.69315	0.22314	3	34
$m) -0.80, -0.70, -0.60, 0.70$	0.35667	0.22314	6	34
$n) -0.80, -0.10, 0.10, 0.70$	0.91629	0.22314	2	34
$o) -0.80, -0.10, 0.10, 0.10$	0.79851	0.22314	3	34
$p) -0.90, -0.60, -0.30, 0.00$	0.79851	0.10536	2	70
$q)\ \ 0.90, 0.85, 0.80, 0.75$	0.22314	0.10536	7	42
$r) -0.90, -0.85, -0.80, -0.75$	0.16252	0.10536	14	70
$s) -0.90, -0.85, -0.80, -0.10$	0.16252	0.10536	14	70
$t) -0.90, -0.10, 0.00, -0.10$	0.79851	0.10536	2	70

In Case 1 where $\nu(X)$ is always equal to one we get fairly rapid convergence for

$-1 < \rho\,(I - GA) < 1$. If any of the eigenvalues have modulus greater than one then, of course, the iteration process does not converge, and if we choose one of the eigenvalues to be equal to one then it converges to an incorrect solution. This is because $\lambda_i = 1$, for some i, implies that $(I - \Lambda)$ has a zero row and column and therefore one of the solutions x_i remains unchanged at each iteration.

Case 1a) shows that if all the eigenvalues are close to zero then the initial and asymptotic rates of convergence are high and as such the iteration process terminates very quickly.

In cases b)–d) three of the eigenvalues are kept constant with the other increasing in magnitude resulting in a slight decrease in the convergence rates and a small increase in the total number of iterations. Cases e)–h) have the eigenvalue with the largest modulus constant while the other three are varied. These variations do not affect the convergence rates or the number of iterations.

Cases j)–o) test whether the distribution of the eigenvalues affects convergence. In each case the smallest and largest eigenvalues are kept constant with the other two spaced out in between, non-uniformly. This results in a small decrease in the initial rate, but the asymptotic rate and the number of iterations are unchanged.

Cases p)–t) show that if all the eigenvalues are close to one, then the initial and asymptotic rates are lower, but where at least half the eigenvalues are closer to zero than one, the asymptotic rate is attained after two iterations like all previous cases.

Repeated eigenvalues [i) and o)], do not slow convergence themselves, it is only their magnitude that is the determinant factor.

In almost all cases, we attain the asymptotic rate in two iterations, so we do have

rapid early convergence, with the actual rate being greater for a set of eigenvalues with lower magnitude.

Case 2

Now we are going to see how the condition number of $\nu(X)$, of X, affects convergence. Since we are able to find a G to assign any eigenvectors to the matrix $(I - GA)$, we can choose X to be any nonsingular 4×4 matrix. By keeping Λ to be constant we can see what effect X, i.e. $\nu(X)$, has.

Let $\Lambda = diag(-0.5, -0.1, 0.3, 0.7)$. Choose nonsingular matrices X which have different condition numbers, calculate G from Equation (2.9), and apply the iteration process.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1.1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

a) $\nu(X) = 1$ b) $\nu(X) = 1.1$ c) $\nu(X) = 2.0$ d) $\nu(X) = 4$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1.5 & 0 & 1.5 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 2 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 2 & 0 \\ 2 & 1 & 1 & 2 \end{pmatrix}$$

e) $\nu(X) = 5$ f) $\nu(X) = 8$ g) $\nu(X) = 10.67$ h) $\nu(X) = 12$

$$\begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 1 & 0 & 3 & 2 \end{pmatrix} \qquad \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 2 & 1 & 2 & 1 \end{pmatrix}$$

i) $\nu(X) = 19.50$ j) $\nu(X) = 50.92$ k) $\nu(X) = 72$

Table 6

Choice of X	*initial rate of convergence*	*asymptotic rate of convergence*	*No. of iter. for asymp. rate*	*total no. of iterations*
a)$\nu(X) = 1.00$	1.04982	0.35667	2	17
b)$\nu(X) = 1.10$	1.04982	0.35667	2	17
c)$\nu(X) = 2.00$	1.04982	0.35667	2	17
d)$\nu(X) = 4.00$	1.04982	0.35667	2	17
e)$\nu(X) = 5.00$	1.04982	0.35667	7	20
f)$\nu(X) = 8.00$	1.04982	0.35667	8	20
g)$\nu(X) = 10.67$	0.75463	0.35667	14	20
h)$\nu(X) = 12.00$	0.62549	$***$	$*$	19
i)$\nu(X) = 19.50$	0.50931	$***$	$*$	19
j)$\nu(X) = 50.92$	0.16252	$***$	$*$	20
k)$\nu(X) = 72.00$	0.00157	$***$	$*$	20

We can see from Table 6 that the iteration process is unaffected for $\nu(X) \leq 4$ and as the condition number increases further, the initial convergence rates are lower and the total number of iterations is greater. For $\nu(X) > 10.7$, the initial rate is much lower and the convergence rates do not converge to the correct asymptotic

rate. Previously, we began with high convergence rates which decreased until reaching the asymptotic rate and then assumed that constant value. When the condition number of X is significantly high, the convergence rates continue to fall even below the asymptotic rate without achieving a constant, uniform value. The * notation represents a number significantly lower than the asymptotic value. The iteration does converge, but is not as fast.

If we now assign eigenvalues of lower magnitude and follow the method above, with the same matrices X, then we get a similar result. Let $\Lambda = diag(-0.10, -0.05, 0.05, 0.1)$.

Table 7

Choice of X	*initial rate of convergence*	*asymptotic rate of convergence*	*No. of iter. for asymp. rate*	*total no. of iterations*
$a)\nu(X) = 1.00$	2.99573	2.30259	2	4
$b)\nu(x) = 1.10$	2.99573	2.30259	2	4
$c)\nu(X) = 2.00$	2.99573	2.30259	2	4
$d)\nu(X) = 4.00$	2.72747	$***$	$*$	5
$e)\nu(X) = 5.00$	2.72747	$***$	$*$	5
$f)\nu(X) = 8.00$	2.72747	$***$	$*$	5
$g)\nu(X) = 10.67$	2.07944	$***$	$*$	5
$h)\nu(X) = 12.00$	2.07944	$***$	$*$	5
$i)\nu(X) = 19.50$	1.89712	$***$	$*$	5
$j)\nu(X) = 50.92$	1.81708	$***$	$*$	5
$k)\nu(X) = 72.00$	1.79263	$***$	$*$	5

Again we find that for low condition numbers convergence is unaffected, but if

the condition number of X is high, then even a set of eigenvalues close to zero do not produce rapid convergence, the * represents a figure significantly lower than the asymptotic rate.

In Case 1, we concluded that the majority of the eigenvalues needed to be close to zero for rapid early covergence. Case 2 demonstrates that, while the above condition may be necessary, it is not sufficient. Therefore, we require both eigenvalues of low magnitude and X, the matrix of the corresponding eigenvectors, to be **well conditioned**, too.

So in Chapter 3, when we apply control techniques to assign eigenvalues and some or all of the eigenvectors, we must aim to assign eigenvalues of as low magnitude as possible which yield a matrix X that is reasonably well conditioned.

Chapter 3

Application of Control Techniques in Iterative Methods

In this chapter we discuss **robust pole assignment**, i.e. a way of assigning eigenvalues and eigenvectors by state feedback in the linear time invariant system. This procedure is generally used in Control Theory to stabilise an unstable system. We are going to apply this procedure to try to ensure rapid convergence of the iteration process. We shall do this by using the pole assignment technique to control the eigenstructure, i.e. the eigenvalues and the eigenvectors, of the iteration matrix so that it is convergent.

The state feedback pole assignment problem in control system design is essentially an inverse eigenvalue problem. A desirable property of any system design is that the poles should be insensitive to perturbations in the coefficient matrices of the system equations. There are many ways of assigning eigenvalues [See Barnett (1975) for an example and further references] but we are looking to find a robust i.e. a **well conditioned** solution.

3.1 Robust pole assignment

We now consider the time invariant, linear, multivariable system :

$$\frac{d\mathbf{x}}{d\mathbf{t}} = A\mathbf{x} + B\mathbf{u} \tag{3.1}$$

where $\mathbf{x}$, $\mathbf{u}$ are $n-$ and $m-$ dimensional vectors, respectively, and A, B are real, constant matrices. Matrix B is assumed to have full rank. The behaviour of system (3.1) is determined by its **poles**, i.e. the eigenvalues of A. In order to modify the poles and make it stable, we choose a state-feedback control, $\mathbf{u}$, given by :

$$\mathbf{u} = F\mathbf{x} + \mathbf{v} \tag{3.2}$$

where the matrix F, the **feedback** or **gain matrix**, is chosen such that the modified dynamic system :

$$\frac{d\mathbf{x}}{d\mathbf{t}} = (A + BF)\mathbf{x} + B\mathbf{v} \tag{3.3}$$

now with an input $\mathbf{v}$, has the desired poles.

The state feedback pole assignment problem for system (3.1) can be formulated as follows :

Problem 1

Given real matrices (A, B) of orders $(n \times n, n \times m)$ respectively, and a set of n complex numbers , Δ=$(\lambda_1, \lambda_2 ..... \lambda_n)$, closed under complex conjugation, find a real $m \times n$ matrix F such that the eigenvalues of $A + BF$ are λ_j, $j = 1, 2, ...n$.

The conditions for a solution to exist are well known and the following theorem is well established (Wonham 1979).

Theorem :

A solution F to Problem 1 exists for **every** set Δ of self conjugate complex numbers if and only if the pair (A, B) is **completely controllable,** that is , if and only if:

$$\mathbf{s}^T A = \mu \mathbf{s}^T \quad and \quad \mathbf{s}^T B = \mathbf{0} \quad \Longleftrightarrow \quad \mathbf{s}^T = \mathbf{0}.$$

If the pair (A, B) is not controllable, i.e. there exists $\mathbf{s}^T \neq \mathbf{0}$ such that $\mathbf{s}^T A = \mu \mathbf{s}^T$ and $\mathbf{s}^T B = \mathbf{0}$, then

$$\mathbf{s}^T (A + BF) = \mu \mathbf{s}^T \quad for \quad all \quad F.$$

Then μ is an eigenvalue of $A + BF$ for all F and must belong to any set Δ of poles to be assigned. The pole μ is said to be **uncontrollable**, and it cannot be modified by any feedback control.

In the single-input case ($m = 1$), the solution to Problem 1, when it exists, is unique. For proof see Mayne and Murdock (1970). If $1 < m < n$, then various solutions exist. Finally, if $m = n$, then a solution always exists, since $rank(B) = n$ implies that the left null space of B contains only the trivial solution and the pair (A, B) is always **completely controllable**; therefore any feedback matrix can always be achieved by feedback, i.e. in this case we can assign the matrix $(A+BF)$ to have any desired eigenstructure.

The main aim in Control Theory is to develop methods for finding F, so that the system is **robust** i.e. insensitive to pertubations. Let $\mathbf{x}_j$ and $\mathbf{y}_j$, $j = 1, 2, ..., n$, be the right and left eigenvectors of the closed loop system matrix $M \equiv A + BF$,

corresponding to eigenvalue λ_j :

$$M\mathbf{x}_j = \lambda_j, \quad \mathbf{y}_j^T M = \lambda_j \mathbf{y}_j^T \tag{3.4}$$

If M is *non-defective*, then M is diagonalizable and it can be shown [Wilkinson (1965)] that the sensitivity of the eigenvalue λ_j to pertubations in A,B and F depends upon the magnitude of the **condition number** c_j, where

$$c_j = 1/s_j = \frac{||\mathbf{y}_j||_2 \, ||\mathbf{x}_j||_2}{\left|\mathbf{y}_j^T \mathbf{x}_j\right|} \geq 1 \tag{3.5}$$

Wilkinson (1965) gives a bound on the sensitivities of the eigenvalues,

$$\max_j c_j \leq \nu(X) \equiv ||X||_2 \left|\left|X^{-1}\right|\right|_2 \tag{3.6}$$

where $\nu(X)$ is the **condition number** of X. Note that the condition numbers take a minimum value $c_j = 1$, for all $j = 1, 2, ..., n$, if and only if M is a normal matrix, that is $M^*M = MM^*$. In this case the eigenvectors of M may be chosen to be an orthonormal basis, and therefore X is perfectly conditioned with $\nu(X) = 1$.

Problem 1 can now be re-written to allow for **robustness** :

Problem 1′

Given (A, B) and Δ (as in Problem 1), find a real matrix F and non-singular matrix X satisfying

$$(A + BF)X = X\Lambda \tag{3.7}$$

where $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_n)$ such that some measure ν of the conditioning, or robustness, of the eigenproblem is optimized.

Note that, in the case where $m = 1$, if F exists then X is uniquely determined

(up to scaling), and therefore the condition numbers, c_j, cannot be controlled. When $m = n$, we can choose X to be orthogonal, (e.g. $X \equiv I$), and hence $c_j = 1, \forall j$. The more interesting cases in Control Theory, which attract more attention, are for the general multi-input system where $1 < m < n$. In these cases, it is possible to control the sensitivities of assigned poles to a restricted extent by an appropriate choice of eigenvectors.

It would be a good idea to ask under what conditions a given nonsingular matrix X can be assigned to the system (3.7). The following theorem helps answer this question;

Theorem :

Given $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_n)$ and X nonsingular, then there exists F a solution to (3.7) **if and only if**

$$U_1{}^T(AX - X\Lambda) = 0 \tag{3.8}$$

where

$$B = [U_0, U_1] \begin{bmatrix} Z \\ \\ 0 \end{bmatrix} \tag{3.9}$$

with $U = [U_0, U_1]$ orthogonal and Z nonsingular. Then F is given explicitly by:

$$F = Z^{-1}U_0^T(X\Lambda X^{-1} - A) \tag{3.10}$$

Proof: [Can also be found in Kautsky and Nichols ((1985)]

The assumption that B is of full rank implies the existence of the decomposition (3.9). Form (3.7), F must satisfy :

$$BF = X\Lambda X^{-1} - A \tag{3.11}$$

and pre-multiplication by U^T then gives the two equations

$$ZF = U_0^T(X\Lambda X^{-1} - A)$$

$$0 = U_1^T(X\Lambda X^{-1} - A)$$

from which (3.8) and (3.10) follow directly, since X is invertible from the condition that X is nonsingular.

The decomposition (3.9) of B can be taken, for example, to be the singular value decomposition (SVD) or the QR decomposition. In general, the SVD method is considerably more expensive than the QR method. With the SVD $Z = \Sigma V^T$, where $\Sigma = diag(\sigma_1, \sigma_2, ..., \sigma_n)$ is a positive matrix and V is orthogonal, and with the QR method Z is an upper-triangular matrix.

3.2 Numerical Algorithm

We now consider the practical implementation of the theory discussed in the previous section for the linear state feedback design. This algorithm is taken from a Numerical Analysis Report by Kautsky and Nichols (1983). The procedure consists of three basic steps :

Step 1:

Compute the decomposition of matrix B by either using SVD or QR, to find the matrices U_0, U_1 and Z , and construct the orthonormal bases, comprised of the columns of the matrices S_j and $\hat{S}_j$ for the null space $S_j = \mathcal{N}[U_1^T(A - \lambda_j I)]$ and its complement $\hat{S}_j$ for $\lambda_j \in \Delta, j = 1, 2, ..., n$.

Standard library software is available to compute the decomposition of B using either SVD or QR.

We consider two methods to find the orthonormal bases S_j and $\hat{S}_j$:

Case 1: QR method

We determine the QR decomposition of $({U_1}^T(A - \lambda_j I))^T$ partitioned as :

$$(U_1^T[A - \lambda_j I])^T = [\hat{S}_j, S_j] \begin{bmatrix} R_j \\ \\ 0 \end{bmatrix} \tag{3.12}$$

Then S_j, and $\hat{S}_j$ are the matrices we require.

Case 2: SVD method

We determine the singular value decomposition of ${U_1}^T(A - \lambda_j I)$ in the form :

$$U_1^T(A - \lambda_j I) = T_j[\Gamma, 0][\hat{S}_j, S_j]^T \tag{3.13}$$

Then the columns of S_j and $\hat{S}_j$ give the required orthonormal bases.

Step 2:

Select vectors $\mathbf{x}_j = S_j \mathbf{w}_j \in S_j$ with $||\mathbf{x}_j|| = 1,\ j = 1, 2, ..., n$, and such that $X = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]$ is **well conditioned**.

There are four basic methods which are discussed by Kautsky and Nichols, out of which I shall only discuss the simplest method because this particular step is not necessary for my application. The iteration matrix is formed in such a way that the matices I and A are both of equal dimension, in which case this step of selecting orthogonal vectors is not necessary. Each of the four methods described in the source paper aim to mimimize a different measure of the conditioning of matrix X. This is done by an iteration where we have an initial set of eigenvectors

$\mathbf{x_j} \in S_j$ and

$$X_j = [\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_{j-1}}, \mathbf{x_{j+1}}, .., \mathbf{x_n}]$$

At each sweep of the iteration, the vector $\mathbf{x_j}$ is replaced by a new vector $\mathbf{x_j} \in S_j$, selected to improve the conditioning of X. A complete sweep has been made when j has run from 1 to n. The iteration is then continued with the new matrix X until it becomes well conditioned in some sense (i.e. the condition number of X becomes unchanged for some tolerance).

Step 3:

Find the matrix $M = A + BF$ by solving $MX = X\Lambda$ and compute F explicitly from $F = Z^{-1}U_0^T(M - A)$.

The matrix $M = X\Lambda X^{-1}$ is constructed by solving the following equation $X^T M^T = (X\Lambda)^T$ for M^T using direct LU decomposition or Gaussian Elimination methods.

A program has been written in MATLAB to carry out the above procedures and assign required eingenvalues to the matrix $(A + BF)$.

3.3 Applications

Before we begin to apply the pole assignment procedure to our problem, there are two important issues which need attention.

The first thing to note is that the numerical algorithm, described above, is used to determine the eigenvalues of the matrix $(A + BF)$, for given matrices A and B. But our problem requires us to assign eigenvalues to the iteration matrix $(I - GA)$, for given matrices I and A. This problem is in fact, in the form of :

Problem 2

Given real matrices (A, C) of orders $(n \times n, m \times n)$ respectively, and a set of n complex numbers , Δ=$(\lambda_1, \lambda_2, ..., \lambda_n)$, closed under complex conjugation, find a real $n \times m$ matrix G such that the eigenvalues of $A - GC$ are $\lambda_j, j = 1, 2, ...n$.

Problem 2 is in fact the dual of Problem 1. Whereas Problem 1 was a controller problem, this is its dual known as an observer problem. This makes very few changes to the numerical algorithm, presented above. Equation (3.7) now becomes :

$$(A - GC)X = X\Lambda \tag{3.14}$$

where G is to be found. A detailed algorithm for eigenvalue assignment to observer systems can be found in the thesis by S. Stringer(1993). But for our problem we can solve the dual problem and obtain the required matrix G. We need to substitute :

$$A = A^T \qquad B = C^T \qquad and \qquad F = -G^T$$

into the original (cotroller) system and solve it using the numerical algorithm. We can then obtain G from it. So to assign eigenvalues to $(I - GA)$, we put $A = I^T = I$, $B = A^T$ and calculate F. Hence :

$$G = -F^T$$

The other note of importance is that due to the formulation of our iteration process, the matrices A and B are both $n \times n$ matrices. This means that $m = n$ and therefore, as discussed in Section (3.1) we can assign any eigenvalues to $(A + BF)$

and choose X such that $\nu(X) = 1$. Because of this, this case does not attract much attention and therefore it is difficult to find test cases for it.

Example 1: [Cavin and Bhattacharyya(1983)]

$$n = 1, \quad m = 2, \quad \Lambda = (-1, -2)$$

$$A = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

This gives

$$X = \begin{bmatrix} -0.635 & -0.591 \\ -0.317 & -0.394 \end{bmatrix}$$

and

$$F = [2.00 - 6.00]$$

These results obtained by pole assignment agree with Cavin and Bhattacharyya, which they obtained using an application of Sylvester's equation, since $m = 1$ and therefore F is unique. $\nu(X) = 15.98$ and cannot be controlled.

Example 2: [Barnett(1975)]

$$n = 3, \quad m = 2, \quad \Lambda = (1, 1, 3)$$

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 6 & -11 & 6 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}$$

This gives

$$F = \begin{bmatrix} -1.6053 & 3.0941 & -1.4887 \\ -2.0941 & 4.2907 & -2.1966 \end{bmatrix}$$

This result is obtained after two iterations and the condition number, $\nu(X) = 7.81$ is the best that can be achieved since the problem is not completely controllable.

Example 3:

$$\tilde{A} = \begin{pmatrix} 10 & -1 & 2 & 0 \\ -1 & 11 & -1 & 3 \\ 2 & -1 & 10 & -1 \\ 0 & 3 & -1 & 8 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 6 \\ 25 \\ -11 \\ 15 \end{pmatrix}$$

We wish to assign $\Lambda = diag(-0.1, -0.05, 0.05, 0.1)$ to the matrix $(I - G\tilde{A})$ and then apply the iteration process with the computed G.

If we substitute $A = I^T = I$ and $B = \tilde{A}^T$ we can find F easily from Equation (3.10), viz.,

$$F = Z^{-1}U_0^T(X\Lambda X^{-1} - A)$$

by taking $X = I$.

Having obtained F, G is found from $G = -F^T$:

$$G = \begin{pmatrix} 1.1557809e-01 & 1.0263692e-02 & -2.2758621e-02 & -6.6937120e-03 \\ 9.7971602e-03 & 1.0762677e-01 & 4.8275862e-03 & -3.9756592e-02 \\ -1.9655172e-02 & 4.3678161e-03 & 1.0045977e-01 & 1.0919540e-02 \\ -5.4766734e-03 & -3.4077079e-02 & 1.0344828e-02 & 1.2657201e-01 \end{pmatrix}$$

This value of G will assign the required eigenvalues to $(I - G\tilde{A})$. If we now use this G in our iteration process :

$$\mathbf{x}^{n+1} = (I - G\tilde{A})\mathbf{x}^n + G\mathbf{b}$$

then the method converges to the exact solution in four iterations. From Examples $1-3$ of Chapter 2 we realised that Jacobi converged in 10 iterations, Gauss-Siedel in 5 and the Preconditioned Conjugate Gradient method in 4 iterations. So for this particular example, we have improved on Jacobi and Gauss-Siedel. Since we can assign any eigenvalues to $(I - G\tilde{A})$ and still achieve $\nu(X) = 1$, we can assign eigenvalues of a smaller magnitude than above and see what the outcome is. If we assign the following eigenvalues :

$$\Lambda = diag(-0.001, -0.0005, 0.0005, 0.001)$$

then we obtain :

$$G = \begin{pmatrix} 1.0507204e-01 & 9.3307221e-03 & -2.0689862e-02 & -6.0852535e-03 \\ 9.3306755e-03 & 1.0250220e-01 & 4.5977241e-03 & -3.7863611e-02 \\ -2.0689552e-02 & 4.5976782e-03 & 1.0574660e-01 & 1.1494195e-02 \\ -6.0851318e-03 & -3.7863043e-02 & 1.1494138e-02 & 1.4063416e-01 \end{pmatrix}$$

and the iteration process with this given G converges after just three iterations. If we now assign $\Lambda = diag(-0.00001, -0.000005, 0.000005, 0.00001)$. Then the computed matrix G enables our iteration to find the exact solution in just two iterations. The values at each iteration are shown below :

Table 8:

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$
0	0.00000000	0.00000000	0.00000000	0.00000000
1	0.99999997	1.99999999	−1.00000004	0.9999999
2	1.00000000	2.00000000	−1.00000000	1.0000000

It can easily be seen that this value of G ensures **very** rapid convergence. The values after just one iteration are very close to the exact solution itself. So for this linear system, the pole assignment procedure is much more efficient than the standard iteration methods discussed in Chapter 2. We cannot improve on this by proceeding with this idea, and assigning eigenvalues of even smaller magnitude, although the the values after one iteration are very, very close to the exact solution.

We now we consider another example, where A is a sparse, unsymmetric 10×10 matrix.

Example 4:

$$\tilde{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 1 & 0 & 4 & 0 & 0 & -1 & 0 & 0 \\ 0 & 3 & 0 & 1 & 0 & 0 & -1 & 0 & 2 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 5 & 0 & 0 & -2 \\ -1 & 0 & 0 & 2 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 1 & -3 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & 1 & 0 \\ 0 & 3 & 0 & 0 & 0 & -2 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 4 & 0 & 0 & -2 & 0 & 1 & 1 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 1 \\ 10 \\ 0 \\ -11 \\ 24 \\ 3 \\ -8 \\ 2 \\ 9 \\ 2 \end{pmatrix}$$

If we choose to assign $\Lambda = diag(-0.1, -0.08, -0.06, -0.04, -0.02, 0.00, 0.02, 0.04, 0.06, 0.08)$, to (A + BF) then the resulting matix G ensures convergence in 4 iterations. If we continue to assign eigenvalues of smaller magnitude then, as in the previous case, we can attain the exact solution in just two iterations.

Chapter 4

Conclusions

We have studied three standard iteration methods for solving the linear system $A\mathbf{x} = \mathbf{b}$, and found, unsurprisingly, that the Preconditioned Conjugate Gradient Method was most efficient.

$$\mathbf{x}^{n+1} = (I - GA)\mathbf{x}^n + G\mathbf{b} \tag{4.1}$$

By analysing the matrix form of the iteration process given by Equation (4.1), we showed that convergence was not only dependent on the eigenvalues of the matrix $(I-GA)$, but also on the corresponding eigenvectors. Therefore an efficient iteration process must combine eigenvalues of low magnitude with eigenvectors which are as orthogonal as possible; yielding a low condition number $\nu(X)$.

In noticing this, I considered an application of Control Theory, to determine the eigenstructure of $(I - GA)$. Since I and A are both of full rank and the pair (I, A) is completely controllable [See Section 3.1], we can find a matrix G to assign any desired eigenvalues to the iteration matrix and obtain a perfectly well-conditioned matrix X. This procedure enables us to find a G that ensures **very** rapid convergence of the iteration process (4.1). This method has been

demonstrated to have very rapid convergence, even in the case of a fairly sparse matrix.

The procedure can be used to solve large linear systems, very speedily. The only conditions necessary for the application of this method are that :

1) the matrix A must be of full rank, and

2) the pair (I, A) must be completely controllable.

The pair (I, A) is **completely controllable** if and only if:

$$\mathbf{s}^T I = \mu \mathbf{s}^T \quad and \quad \mathbf{s}^T A = \mathbf{0} \quad \Longleftrightarrow \quad \mathbf{s}^T = \mathbf{0}.$$

If the problem is not completely controllable then we may still assign the eigenvalues but the choice of corresponding eigenvectors will be restricted, and therefore X may not be well-conditioned. Nevertheless we can still apply the numerical algorithm. Step 2 of the algorithm selects the eigenvectors from the required subspace, in order to minimise $\nu(X)$ – see Example 2 in Chapter 3. This procedure may help improve the rate of convergence, depending on the condition number of X.

Bibliography

[1] Barnett S. *Introduction to Mathematical Control Theory.* Oxford University Press, Oxford (1975).

[2] Burden R.L. and Faires J.D. *Numerical Analysis 3rd Edition.* Prindle, Weber and Schmidt, Boston (1985).

[3] Cavin R.K. and Bhattacharyya S.P. *Robust and Well-conditioned Eigenstucture Assignment Via Sylvester's Equation.* Optimal Control Applics. and Methods, Vol.4, 205-212 (1983).

[4] Golub G.H. and Van Loan C.F. *Matrix Computations.* The Johns Hopkins University Press (1983).

[5] Kautsky J. and Nichols N. *Robust Eigenstructure Assignment.* Numerical Analysis Report (1983).

[6] Kautsky J., Nichols N. and Van Dooren P. *Robust Pole Assignment in Linear Feedback.* Int. J. Control, Vol.41, No.5, 1129-1155 (1985).

[7] Ogata K. *State Space Analysis of Control Systems.* Prentice-Hall (1969).

[8] Stringer S.M. *The Use of Robust Observers in the Simulation of Gas Supply Networks.* Reading University Thesis (1993).

[9] Varga R.S. *Matrix Iterative Analysis.* Prentice-Hall (1962).

[10] Wilkinson J.H. *The Algebraic Eigenvalue Problem.* Oxford University Press, Oxford (1965).

[11] Wonham W.M. *On Pole Assignment in Multi-input Controllable Systems.* IEEE Trans. Auto. Control AC-12, 660-665 (1967).

[12] Young D.M. *Iterative Solution of Large Linear Systems.* Academic Press, New York (1971).