Numerical solution of Markov Chains

Amr Lotfy Elsayad

NUMERICAL SOLUTION OF MARKOV CHAINS

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Arts

in

Mathematics

by

Amr Lotfy Elsayad

March 2002

NUMERICAL SOLUTION OF MARKOV CHAINS

A Project

Presented to the

Faculty of

California State University,

San Bernardino

by

Amr Lotfy Elsayad

March 2002

Approved by:

[redacted] 2/8/02

Terry Hallett, Chair, Mathematics Date

[redacted]

John Sarli

[redacted]

Chetan Prakash

[redacted]

Peter Williams, Chair
Department of Mathematics

Terry Hallett,
Graduate Coordinator
Department of
Mathematics

ABSTRACT

This project deals with techniques to solve Markov Chains numerically. It acquaints the reader with Markov Chains and their applications in the first chapter. Then it discusses the classical Gaussian Elimination for solving Markov Chains in the second chapter. Chapter three introduces the reader to iterative methods including the common Jacobi and Gauss Seidel methods. Convergence of a general iterative scheme is also discussed. The problem of slow convergence is illustrated by an example. Methods of speeding up convergence however are not discussed. The fourth chapter is probably the most important chapter of the project as it describes relatively recent techniques developed to solve linear systems with sparse matrices like those found in Markov Chains. These techniques are called projection methods. To this end, a prototype projection step is given and two common algorithms, explained. Finally, the concluding chapter summarizes and sheds light on the advantages and disadvantages of the different methods used in this project.

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER ONE

INTRODUCTION

Overview

Linear equations arise in a lot of engineering and scientific applications. One of these applications is computing the probability distribution of a Markov chain at the steady state. A Markov chain is a system whose states change such that the probability of the system to be in a certain state depends only on the state prior to it. This project aims at identifying the problem and describing how the specific structure of the coefficient matrix directed mathematicians to look for efficient methods for solving Markov chains numerically. A standard notation for a general system of linear equations is $Ax = b$. When it comes to Markov chains, we usually seek a system of the form $\pi P = \pi$, where P is a stochastic matrix and π is the probability distribution at the steady state. In other words $\sum_{\text{all } j} p_{ij} = 1$ for $i = 1,...,n$, and $\|\pi\|_2 = 1$. So, the goal is to find the eigenvector that corresponds to the unit eigenvalue.

An overall view of Markov chains is given in the first part of this study. Discrete and continuous Markov chains are introduced. Then, the steady state equations are derived in both cases. The embedded Markov chain is also discussed and is illustrated by an example. A queuing model as an example of a continuous Markov chain concludes the introduction.

Direct methods are then introduced by explaining a common one, Gaussian elimination. Then, some iterative methods are introduced such as the power method and Gauss Seidel's method. The study also presents the problem of slow convergence rates that arises when the ratio between the dominant eigenvalue and the subdominant is approximately equal to one. The Courtois matrix is an example of a matrix that exhibits this property of slow convergence.

Finally projection methods are discussed. Projection methods are relatively recent and have proved efficient in solving Markov chains. A general projection scheme is presented in this paper along with two methods: Arnoldi's method and the generalized minimal residual's method, GMRES. Both methods require finding an orthonormal basis for the Krylov subspace. To this end Arnoldi's process is presented and demonstrated by an example. The strength of the GMRES method lies in the fact that it can be used in solving nonsymmetric linear systems. If the given linear system is positive definite, then it is recommended to use the well-known conjugate gradient method.

Markov Chains

Stochastic Processes describe phenomena such as the weather of a city on a given day, which can fall into one of several states .For example, if we assumed that the weather could only be cloudy, partly cloudy, or sunny, then it would be a point of interest to know the probability of having three consecutive sunny days. In another instance, we would be interested in knowing the probability of having one cloudy day followed by a sunny day. The preceding scenario is an example of a stochastic process. In general, stochastic processes can be described by$\{X(t), t \in T\}$, where $X(t)$ is a family of random variables indexed by a parameter t. The set of values that $X(t)$ can take on is called the state space. If those values were continuous, as in the case of the level of water in a dam, then the stochastic process would be called a continuous state stochastic process. Otherwise, the state would be considered a discrete state stochastic process. If the parameter t was continuous then the process would be called a continuous time stochastic process. Otherwise, the process would be called discrete time stochastic process. In the context of stochastic processes, the three different weather conditions form what is called the state space. As such, the weather condition on a given day may be described by a discrete random variable that takes on three values. This set of random variables form a stochastic process. As in many cases the random variables are indexed by the time T. In the above example, T would take only discrete values representing the days of the week. As such, the stochastic process in hand can be described by{X(t), t $\in$days of the week}, where X(t) represents the weather condition on a given day. Then, for example, T would have the form $T=\{t : 0 \leq t \leq +\infty\}$.

A stochastic process that does not change when an arbitrary shift of time is introduced is called a stationary stochastic process. An example of discrete space stochastic process could be the number of planes that crash in the US during a given year. Examples of continuous state space stochastic processes are the level of water in a dam and the temperature inside a nuclear reactor. A Markov process is a stochastic process with a conditional probability function satisfying the so-called Markov property described below. A continuous time Markov process is a stochastic process indexed by a continuous parameter t with a discrete state space. Moreover its conditional probability distribution function has the " Markov Property," i.e.,

$$\mathrm{Prob}\ \{X(t) \leq x \,|\, X(t_0) = x_0, X(t_1) = x_1, \ldots\ldots, X(t_n) = x_n\}$$

$= \mathrm{Prob}\ \{X(t) \leq x \,|\, X(t_n) = x_n\}$ for any sequence $t_0, t_1, \ldots, t_n, t$ such that $t_0 < t_1 < \ldots. < t_n < t$. More simply put, the current state depends only on the state at time t_n. Therefore, the system states prior to t_n have no effect on the current state at time t. The time spent in a state is called the sojourn time. In order for the Markov property to be satisfied, the time spent in a state should not affect the remaining time that will be spent in that state. For this to happen the time spent in a given state has to follow an exponential distribution if the Markov process at hand is continuous. If the Markov process were discrete, then the distribution of the time spent in a given state would have to be geometric. If the transition from one state to another depends on the time the transition occurs, then the Markov process is said to be non-homogenous.

Discrete Markov Chains

A Markov chain is said to be a discrete time Markov chain if the time parameter can take on only discrete values. Therefore, the stochastic process at hand would consist of the random variables $X(0), X(1), X(2), \ldots, X(n)$. The values are the states of the process at time t, where $t \in \{0, 1, 2, \ldots\}$. Assume $p_{ij} = \text{prob}\{x_{n+1} = j \mid x_n = i\}$. The matrix $P = \{p_{ij}\}$ is called the transition probability matrix.

The Chapman-Kolmogorov Equations:

$p_{ij} = \text{prob}\{x_{n+1} = j \mid x_n = i\}$ is a single-step transition probability. Now it would be appropriate to find an expression for a multiple-step transition probability.

Proposition 1: $p(A \cap B/C) = P(A/B \cap C).P(B/C)$

Proof: $p(A \cap B/C) = P(A \cap B \cap C)/P(C)$

$$p(A/B \cap C).P(B/C) = \frac{p(A \cap B \cap C)}{P(B \cap C)} \times \frac{P(B \cap C)}{P(C)} = p(A \cap B/C)$$

Claim: $p_{ij}^{(n)} = \sum_{\text{all } k} p_{ik}p_{kj}^{(n-1)}$, where $p_{ij}^{(n)}$ is the probability of going from state i to state j in n steps.

Proof: $p_{ij}^{(n)} = \text{p r ob}\{X_n = j \mid x_0 = i\}$

$= \sum_{\text{all } k} \text{prob}\{X_n = j, X_1 = k \mid x_0 = i\}$, $0 < 1 < n$

$= \sum_{\text{all } k} \text{prob}\{X_n = j \mid X_1 = k, x_0 = i\}. \text{Prob}\{X_1 = k \mid X_0 = i\}.$

(By Proposition 1)

By the Markov Property we get

$p_{ij}^{(n)} = \sum_{\text{all } k} \text{prob}\{X_n = j \mid X_1 = k\}. \text{Prob}\{X_1 = k \mid X_0 = i\}$

Now $P_{ij}^{(n)} = \sum_{\text{all } k} p_{kj}^{(n-1)} p_{ik}^{(1)}$, for $0 < l < n$.

In matrix notation, we have $P^{(n)} = P^{(1)}P^{(n-1)}$.

In particular, $P^{(n)} = P^{(1)}P^{(n-1)} = P^n$.

Definition: A state is said to be recurrent if the probability that it will occur again is 1.If the probability that a state will not happen again is positive, then the state is said to be transient. Now let $f_{jj}^{(n)}$ denote the probability that the first return to state j occurs n steps after leaving it. Hence,

$f_{jj}^{(n)} = \text{Prob}\{X_n = j, X_{n-1} \neq j, \ldots, X_1 \neq j \mid X_0 = j\}$, for n =1,2...

Recursively, $P_{jj}^{(n)} = \sum_{l=1}^{n} f_{jj}^{(l)} P_{jj}^{(n-l)}$, $n \geq 1$. Also let f_{jj} be the probability that the system returns to state j at some time, so that $f_{jj} = \sum_{n=1}^{\infty} f_{jj}^{(n)}$. When f_{jj} is equal to 1 then state j is recurrent. In other words, eventually the system will definitely return to state j. Otherwise, the state is said to be transient.

Definition: The mean recurrence time, $M_{jj} = \sum_{n=1}^{\infty} n f_{jj}^{(n)}$. If M_{jj} is finite then state j is said to be a positive recurrent state. If, however, M_{jj} is infinite, then state j is said to be a null-recurrent state. If the Markov chain in hand is finite then none of its states can be null-recurrent. Then, the states are either positive recurrent or transient. Moreover, there exists at least one positive recurrent state.

Definition: A state j is said to be periodic with period k if when leaving j a return would require a multiple of p steps. As such p is the greatest common divisor of the integers n such that $P_{jj}^{(n)} > 0$. If p is equal to 1 then the state is said to be aperiodic.

Thus, $\varphi D_Q^{-1} = \left(\frac{1}{6}, \frac{1}{3}, \frac{1}{3}, \frac{1}{6}\right)\begin{pmatrix} \frac{1}{4} & 0 & 0 & 0 \\ 0 & \frac{1}{6} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \left(\frac{1}{24}, \frac{1}{18}, \frac{1}{12}, \frac{1}{6}\right)$

$\pi = \frac{-\varphi D_Q^{-1}}{\left\|\varphi D_Q^{-1}\right\|_1} = (.12, .16, .24, .48)$. Note that $\pi Q = 0$.

A Queuing model is a common example of a CTMC. For instance, we could test the effect of adding an additional CPU to a computer network. When the size of the state space is small, the calculation of the steady state probabilities is simple. However, in many models the state space is large and hence efficient numerical techniques would be needed to solve such Markov chains. The following is a queueing model as another example of a continuous time Markov chain.

Example:

Suppose we have two stations in tandem. In other words the customer is served by the first station before being served by the second station. We need to make the following assumptions:

The arrival of customers is a Poisson process with parameter λ. There is only one server at each station. Customers are served with an exponentially distributed service time whose parameter is μ. Scheduling is done on a first-come-first-served basis. Customers are treated the same. Thus, we have one class of customers. Both queues have infinite capacities. The states are the ordered pairs (i, j), where i and j are the number of customers waiting in the two queues. Queuing models are not the only field where Markov chains are useful. Computer networks, computer design, and biology are also major fields in which Markov chains could be used to measure the efficiency of their models. These models have developed in a manner that gave rise to a large increase in their state space. Solving linear equations is a very important branch of numerical analysis. The characteristics of Markov chains narrow down the scope of the methods used to solve them numerically. One characteristic, for example, is the fact that the dominant eigenvalue of a stochastic matrix is the unity. This makes the problem of finding the steady state distribution of an ergodic Markov chain equivalent to finding the eigenvector that corresponds to the unit eigenvalue, as we will see later. In this paper, the focus will be on demonstrating some characteristics of Markov chains like the one mentioned above and how they lead us to efficient numerical methods to solve them. Three numerical methods are discussed: direct methods, iterative methods and projection methods. It is worth mentioning that there are other numerical techniques tailored to solving periodic Markov chains and finding transient states. Those techniques could be a good material for further study and could be found in a textbook like [1].

CHAPTER THREE

ITERATIVE METHODS

Iterative methods are commonly used when the linear system in hand is sparse. These systems usually arise when solving differential equations. They also arise in structural analysis where the forces exerted on a truss, for example, are unknowns. Usually, the problem set up is such that only few forces are involved at each joint. Hence, only few unknowns are involved in each equation, which translates into a sparse coefficient matrix. The same situation occurs when trying to solve discrete or continuous Markov chains where a lot of the transition probabilities or transition rates are zeros.

The General Problem

The goal again is to solve $Ax = b$, where A is an $n \times n$ matrix, $x \in R^n$. We are going to use the general fixed point scheme to solve this system. A typical iteration has the form $x^{(k+1)} = \varphi(x^{(k)}), k \in N, \varphi$ is a mapping from R^n to R^n. $Ax = b$ can be written as $x = (I-A)\, x + b$. Let $C = I-A$.

$\varphi(x) = Cx + b$ is called the iteration function. If ξ is a fixed point of $x = \varphi(x)$, i.e., if $\xi = \varphi(\xi)$, then we have

$$x^{(k+1)} - \xi = \varphi\left(x^{(k)}\right) - \varphi\left(\xi\right) = \left(x^{(k)} - \xi\right) = C^k\left(x^1 - \xi\right).$$

If $x^1 \neq \xi$, then the sequence $x^{(k+1)} = \varphi(x^k)$ converges to ξ when $\lim_{k\to\infty} C^{(k)} = 0$. This occurs if and only if the spectral radius of C is strictly less than 1. I will prove only one direction of the above result.

Proof: Assume the contrary, i.e., let $\rho(C) \geq 1$. Therefore there exists an eigenvalue λ with $|\lambda| \geq 1$ and a vector $x \neq 0$ such that $Cx = \lambda x$. Thus, $C^k x = \lambda^k x$.

But $\lim_{k\to\infty} \lambda^k \neq 0$. Hence, $\{C^k\}$ can not be a null sequence.

The Power Method

The power method is an example of an iterative method. In many situations we are interested in finding the eigenvalue with largest or smallest absolute value and their corresponding eigenvectors. The power method is useful in achieving this goal. Actually, the power method can be used to find all eigenvalues and eigenvectors of a diagonalizable matrix as can be seen in [2]. Again when it comes to applying the power method to $\pi P = P$, where P is a transition matrix, we know that the dominant eigenvalue is one. But the power method is sometimes applied to matrices other than P, where the dominant eigenvalue is not one as is the case with the Jacobi and Gauss Seidel methods.

Let A be a diagonalizable $n \times n$matrix. Let $z^{(0)} \in R^n$

Now let's investigate the iteration $z^{(k)} = Az^{(k-1)}, k = 1, 2, ..$

$z^{(k)} = Az^{(k-1)} \Rightarrow z^{(k)} = A^k z^{(0)}$. Let $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$ be the eigenvalues of A. Moreover, let $\{x^1, x^2, \ldots, x^n\}$ be the set of corresponding eigenvectors. Now we can write

$z^{(0)} = \alpha_1 x^1 + \alpha_2 x^2 + \ldots\ldots\alpha_n x^n$. Therefore,

$$z^{(k)} = A^k[\alpha_1 x^1 + \alpha_2 x^2 + \ldots\ldots\alpha_n x^n]$$

$$= \alpha A^k{}_1 x^1 + \alpha_2 A^k x^2 + \ldots\ldots\alpha_n A^k x^n$$

Hence, $z^{(k)} = \alpha_1 \lambda_1{}^k x^1 + \alpha_2 \lambda_2{}^k x^2 + \ldots\ldots\alpha_n \lambda_n{}^k x^n$, since $A^k x_i = \lambda_i{}^k A^k$.

$$= \lambda_1{}^k[\alpha_1 x^1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k x^2 + \ldots\ldots\alpha_n \left(\frac{\lambda_n}{\lambda_1}\right)^k x^n]$$

Now if z_0 is selected so that $\alpha_1 \neq 0$, then we have the following:

If $|\lambda_1| > |\lambda_2|$, then $\lim\limits_{k\to\infty} \frac{1}{\lambda_1{}^k} z^{(k)} = \alpha_1 x^1$.

(Note that when $\lambda_1 = 1$, the above equation becomes $\lim\limits_{k\to\infty} z^{(k)} = \alpha_1 x^1$)

Therefore, $\lim_{k\to\infty}\begin{bmatrix} z_1^{(k)}/\lambda_1^k \\ z_2^{(k)}/\lambda_1^k \\ . \\ . \\ z_n^{(k)}/\lambda_1^k \end{bmatrix} = \begin{bmatrix} \alpha_1 x_1^1 \\ \alpha_1 x_2^1 \\ . \\ . \\ \alpha_1 x_n^1 \end{bmatrix}$

So, $\lim_{k\to\infty} \frac{z_\nu^{(k)}}{\lambda_1^k} = \alpha_1 x_\nu^1, \nu = 1,2,\ldots\ldots,n$ (*)

$\lim_{k\to\infty} \frac{z_\nu^{(k-1)}}{\lambda_1^{k-1}} = \alpha_1 x_\nu^1, \nu = 1,2,\ldots\ldots,n$ (**)

Dividing (*) by (**) yields

$$\lim_{k\to\infty} \frac{z_\nu^{(k)}}{\lambda_1 z_\nu^{(k-1)}} = 1, x_\nu^1 \neq 0, z_\nu^{(k-1)} \neq 0$$

$$\lim_{k\to\infty} \frac{z_\nu^{(k)}}{z_\nu^{(k-1)}} = \lambda_1, x_\nu^1 \neq 0, z_\nu^{(k-1)} \neq 0.$$

Let $q_\nu^k = \frac{z_\nu^{(k)}}{z_\nu^{(k-1)}}, z_\nu^{(k-1)} \neq 0$. Then, $\lim_{k\to\infty} q_\nu^{(k)} = \lambda_1$.

Again the focus is on solving $\pi P = P$, where P is the transition matrix. As the spectral radius of P is equal to the unity, we can claim that $\lambda_1=1$. Hence, $\lim_{k\to\infty} z_k = \alpha_1\pi$, which can be normalized to find π.

Example:

We will apply the power method to a stochastic matrix A with three different initial states. The eigenvalues of A are $\lambda_1 = 1, \lambda_{2,3} = -0.25 \pm 0.5979i$ with corresponding 2-norm of

1, 0.65, respectively. So, $\frac{\lambda_1}{\lambda_2} \approx 0.65$. The approximate eigenvector is accurate to 4 decimal places after 25 iterations because $0.65^{25} \approx 2 \times 10^{-5}$. The MATLAB code for the power method, which is given below, is followed by the output. z0 is the initial vector, and z is the approximate eigenvector that corresponds to the dominant eigenvalue of the given transition matrix A.

```
function z= PM (A,z0)

global A

global z0

global z

z= z0;

for i = 1:20

z=z*A;

end
```

$$A = \begin{bmatrix} 0 & .8 & .2 \\ 0 & .1 & .9 \\ .6 & 0 & .4 \end{bmatrix}$$

Input:

$z_0 = [1, 0, 0]'$

Output:

```
z = [.2813,.2499,.4688]'
```

The same vector z is obtained if the power method is applied to $z_0 = [0,1,0]'$ or $z_0 = [0,0,1]'$

Now let us look at some of the facts related to the establishment of three other common methods, the Jacobi, Gauss Seidel, and SOR methods.

Theoretical Background

Let $Ax = b$. Assume that $A = M-N$ where M is non-singular. $M-N$ is called a splitting of A. Thus, $(M-N)\ x = b$

$Mx = Nx + b \Rightarrow x = M^{-1}Nx + M^{-1}b$ which gives thus the iterative procedure: $x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b$.

Let $H = M^{-1}N \Rightarrow x^{(k+1)} = H\,x^{(k)} + c$, where $c = M^{-1}b$.

H is called the iteration matrix .The iteration matrix is the one that characterizes different iterative techniques such as the method of Jacobi and Gauss Seidel.

The Jacobi Method

The Method of Jacobi can be applied to a non-homogenous system but this would result in a slow rate of convergence. Usually, faster convergence can be obtained when applying the method to homogenous systems. Therefore the goal is to solve $\pi Q=0$ or equivalently $Q^T\pi^T = 0$, where Q is the infinitesimal generator matrix. For simplicity, let $x = \pi^T$ and $Q^T = D - (L+U)$, where D is a diagonal matrix whose entries are the entries are those of the diagonal of Q^T. L and U are strictly lower and upper triangular matrices respectively. To match the method with the general splitting technique explained earlier, we let $M = D$, $N = L+U$. Thus, the iteration matrix H_J is equal to $M^{-1}N$ or equivalently H_J is equal to $D^{-1}(L + U)$. We note that D is non-singular since $d_{ii} \neq 0$ for all i.

This leads to the iterative scheme $x^{(k+1)} = H_J x^{(k)}$ or $x^{(k+1)} = D^{-1}(L + U)x^{(k)}$. Let $s_{ij} = l_{ij} + u_{ij}$. Clearly, $s_{ii} = 0$, for all i. Let $p_{ij} = \sum_{k=1} d_{ik}^{-1}s_{kj}$ Therefore, $p_{ij} = \frac{1}{d_{ii}}s_{ij}$, since $c_{ik} = 0$ if $i \neq k$, where $[c_{ij}] = D^{-1}$. Thus,

$$x_i^{(k+1)} = \frac{1}{d_{ii}}\left(\sum_{j\neq i}(l_{ij} + u_{ij})\,x_j^{(k)}\right).$$

The Gauss Seidel's method is similar to the Jacobi's method. The difference is that the iteration matrix of the method of Gauss Seidel is $(D - L)^{-1}U$. This is equivalent to using $x_i^{(k+1)}$, where $i < j$, to compute $x_j^{(k+1)}$.

Example:

We seek to find the steady state solution of the reliability model whose infinitesimal generator is a 9×9 matrix adapted from [1]. To this end, the Gauss Seidel method is used. The input of the algorithm consists of an approximate solution, the maximum number of iterations, and the infinitesimal generator Q. The output is the approximate solution obtained by the last iteration and the associated norm of the residual vector. The output also includes the iteration matrix B and its eigenvalues. Note that the subdominant eigenvalue is 0.9827, very close in modulus to the dominant eigenvalue. This translates into slow rate of convergence. Setting itmax, the maximum number of iterations, to 800 we obtains an accuracy of six decimal places. If we set itmax = 10, the accuracy gets extremely poor as shown below.

$$A=\begin{pmatrix} -60.4 & 60 & 0 & .5 & 0 & 0 & 0 & 0 & 0 \\ 60 & -90.4 & 120 & 0 & .5 & 0 & 0 & 0 & 0 \\ 0 & 30 & -120.4 & 0 & 0 & .5 & 0 & 0 & 0 \\ .4 & 0 & 0 & -60.7 & 60 & 0 & 1 & 0 & 0 \\ 0 & .4 & 0 & 60 & -90.7 & 120 & 0 & 1 & 0 \\ 0 & 0 & .4 & 0 & 30 & -120.7 & 0 & 0 & 1 \\ 0 & 0 & 0 & .2 & 0 & 0 & -61 & 60 & 0 \\ 0 & 0 & 0 & 0 & .2 & 0 & 60 & -91 & 120 \\ 0 & 0 & 0 & 0 & 0 & .2 & 0 & 30 & -121 \end{pmatrix}$$

x_0 = [.1,.1,.1,.1,.1,.1,.1,.1,.1]',

b=[0,0,0,0,0,0,0,0,0]',and itmax=10

Output:

x=[.2049,.2052,.0514,.2255,
.2263,.0568,.1675,.1665,.0414]'

res=[14.5884,0.3779,0.3302,.3189,
.3082,.2979,.2880,.2784,.2693,.2605]'

Block Iterative Methods

We will illustrate how block iterative techniques can be used to find the steady state probability of an infinitesimal generator matrix. We will also show, by an example that block iterative techniques are generally faster than their iterative counterparts.

First we partition the vector π into N subvectors and the matrix Q into N^2 to obtain block representation of πQ = 0 as follows.

$$(\pi_1, \pi_2, \ldots\ldots, \pi_N)\begin{pmatrix} Q_{11} & Q_{12} & \cdot & \cdot & \cdot & Q_{1N} \\ Q_{21} & Q_{22} & \cdot & \cdot & \cdot & Q_{2N} \\ \cdot & \cdot & \cdot & & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & & \cdot & \cdot \\ Q_{N1} & Q_{N2} & \cdot & \cdot & \cdot & Q_{NN} \end{pmatrix} = 0$$

Then, we split the matrix Q^T by the block splitting:

$Q^T = D_N - (L_N + U_N)$. Here D_N is a block diagonal matrix while L_N and U_N are strictly lower and upper block triangular matrices respectively.

Thus, $D_N = \begin{pmatrix} D_{11} & 0 & . & . & . & 0 \\ 0 & D_{22} & . & . & . & 0 \\ . & . & . & & & . \\ . & . & & . & & . \\ . & . & & & . & . \\ 0 & 0 & . & . & . & D_{NN} \end{pmatrix}$, $L_N = \begin{pmatrix} 0 & 0 & . & . & . & 0 \\ L_{21} & 0 & . & . & . & 0 \\ . & . & . & & & . \\ . & . & & . & & . \\ . & . & & & . & . \\ L_{N1} & L_{N2} & . & . & . & 0 \end{pmatrix}$, and

$$U_N = \begin{pmatrix} 0 & U_{12} & . & . & . & U_{1N} \\ 0 & 0 & . & . & . & U_{2N} \\ . & . & . & & & . \\ . & . & & . & & . \\ . & . & & & . & . \\ 0 & 0 & . & . & . & 0 \end{pmatrix}$$

Similar to the Gauss Seidel method the block Gauss Seidel method is given by the iteration: $(D_N - L_N)x^{(k+1)} = U_N x^{(k)}$. The following example shows the advantage of the block Gauss Seidel method over the Gauss Seidel method with respect to the speed of convergence. The transition matrix was obtained from Courtois [1]. The Block Gauss-Seidel MATLAB function is given a stochastic matrix, and a vector ni which describes the length of each block. I have chosen ni = [3,2,3]. Another two input parameters are the number of outer iterations labeled itmax1 and the number of inner operations itmax2. itmax1 is simply the number of times the program acts on the entire set of blocks, while itmax2 is the number of times the Gauss-Seidel algorithm is called to solve an individual block. The program can be modified in such a way that would enable it to solve the individual blocks using a direct method rather than the Gauss-Seidel procedure. The output of the program, which consists of the solution vector x and the residual vector obtained after 10 iterations, along with the Courtois matrix are given below.

$$P^T = \begin{pmatrix} .85 & .1 & .1 & 0 & .0005 & 0 & .00003 & 0 \\ 0 & .65 & .8 & .0004 & 0 & .00005 & 0 & .00005 \\ .149 & .249 & .0996 & 0 & .0004 & 0 & .00003 & 0 \\ .0009 & 0 & .0003 & .7 & .399 & 0 & .00004 & 0 \\ 0 & .0009 & 0 & .2995 & .6 & .00005 & 0 & .00005 \\ .00005 & .00005 & 0 & 0 & .0001 & .6 & .1 & .1999 \\ 0 & 0 & .0001 & .0001 & 0 & .2499 & .8 & .25 \\ .00005 & .00005 & 0 & 0 & 0 & .15 & .09990 & .55 \end{pmatrix}$$

$$X=\begin{pmatrix}.08928265275448\\.09275763750511\\.04048831201636\\.15853319081979\\.11893820690415\\.12038548110608\\.27779525244933\\.10181926644470\end{pmatrix},\ \text{Residue} = .00000000000303\times10^{-5}$$

CHAPTER FOUR

PROJECTION METHODS

Projection methods are relatively recent techniques aimed at solving systems of linear equations. Again we will be dealing with large and sparse systems. A typical projection method would attempt to find an approximate solution in a subspace of a much lower dimension than that of R^n. The dimension of that subspace will be denoted by m. We will need another subspace to introduce constraints that are necessary to find an approximate solution. Once an approximate solution is found, the process is repeated again until a sufficiently small residual is obtained. The following is a general outline of the method.

Let $Ax^* = f$. (i)

Let K_m and L_m be two subspaces of R^n.

Two common choices for L_m are $L_m = K_m$ and $L_m = AK_m$.

Let $U = \{v_1, v_2, \ldots\ldots, v_m\}$ be a basis of

K_m and $W = \{w_1, w_2, \ldots\ldots, w_m\}$ be a basis of L_m.

Let x_0 be an initial approximation to the solution of (i).

Also, let x be the new approximation obtained by a single projection step such that $x = x_0 + z, z \in K_m$.

Now let us impose the so-called orthogonality condition $\langle f-Ax, v \rangle = 0 \ \forall \ v \in L_m$. Also, assume that $z = V_m y$, where V_m is the matrix whose columns are the vectors of U, and the initial residue be $r_0 = f - Ax_0$. If x were to be the solution of (i), then the following would be true:

$A(x_0 + z) = f$.

$Ax_0 + Az = f$.

Thus, $Az = f - Ax_0$.

Because $\langle f-Ax, v \rangle = 0$,

it follows that $\langle f-A(x_0 + z), v \rangle = 0$.

Hence, $\langle r_0 - Az, v \rangle = 0$, and

$W_m^T(r_0 - AV_m y) = 0$.

Assuming that $W_m^T A V_m$ is nonsingular, we have:

$[W_m^T A V_m]^{-1} W_m^T r_0 = y$, and $x = x_0 + V_m y = x_0 + [W_m^T A V_m]^{-1} W_m^T r_0$

The above is called Petrov-Galerkin approximation.

How do we choose L_m? A good choice is the subspace defined by $L_m = AK_m$. This can be justified by the following theorem.

Theorem 1: $L_m = AK_m$, x is the approximate solution provided by the Petrov-Galerkin approximation $\Leftrightarrow$ it minimizes the Euclidean norm of the residual vector f- Ax, for all

$x \in x_0 + K_m$. Note that it can be shown that $x_0 + K_m$ is a subspace. It is called an affine subspace.

Proof: We will prove only one direction.

Let x^* be any vector in $x_0 + K_m$. Therefore,

$$\|f - Ax^*\|^2 = \|f - A[(x^* - x) + x]\|^2$$

$$= (f - A[(x^* - x) + x, f - A[(x^* - x) + x])$$

$$= (f - Ax, f - Ax) - 2(f - Ax, A(x^* - x)) + (A(x^* - x), A(x^* - x))$$

By the orthogonality condition, the middle term in the above expression is zero.

Thus, $\|f - Ax^*\|^2 = (f - Ax^*, f - Ax^*) + (A(x - x^*), A(x - x^*)) \geq \|f - Ax^*\|^2$

A lot of the projection methods make use of the so-called Krylov subspace defined by:

$K_m(A, v) = \text{span}\{v, A^2v, A^3v, \ldots\ldots, A^{m-1}v\}$ for some $v \in R^n$.

These methods require obtaining an orthonormal basis for the Krylov subspace. One classical method is the known Gram-Schmidt Process.

Gram-Schmidt Orthogonalization Procedure

The input is $X = \{x_1, x_2, \ldots\ldots, x_m\}, x_j \in R^n$

1. Let $r_{11} = \|x_1\|_2$ and $q_1 = \frac{x_1}{r_{11}}$

2. For $j = 2, 3, \ldots\ldots, m$ do

- $r_{11} = q_i^T x_j, i = 1, 2, \ldots, j - 1$
- $q_j = x_j - \sum_{i=1}^{j-1} r_{ij} q_i$
- $r_{jj} = \|q_j\|_2$ and $q_j = q_j / r_{jj}$

The input of the algorithm is an $(n \times m)$ matrix X of rank m less than or equal to m. The columns of X represent the vectors that span a given subspace. Basically, the algorithm factors the matrix X into QR. Q is an $n \times m$ orthogonal matrix whose columns form the basis of the given subspace. R is an $n \times n$ upper triangular matrix. The columns of X can be written as linear combinations of the columns of Q.

Examples:

$$x_2 = q_2 + r_{12} q_1$$
$$x_3 = q_3 + r_{13} q_1 + r_{23} q_3$$

The last line of step 2 ensures that the q_j's are normalized. Now let us prove that Q is orthogonal.

$$q_j = x_j - \sum_{i=1}^{j-1} \left(q_i^T x_j\right) q_i$$

For all $k \le j - 1$, we have

$$q_k^T q_j = q_k^T x_j - \sum_{i=1}^{j-1} q_k^T (q_i^T x_j) q_i$$

$$= q_k^T x_j - \sum_{i=1}^{j-1} q_k^T q_i (q_i^T x_j)$$

$$= q_k^T x_j - q_k^T x_j = 0$$

Gram-Schmidt Orthogonalization Procedure was modified to yield a more numerically stable version than the one mentioned above. The new version was referred to as Modified Gram-Schmidt Orthogonalization Procedure. When the Modified Gram-Schmidt Orthogonalization Procedure is applied to a Krylov subspace the method is referred to as Arnoldi's process. The following is the MATLAB program used to find an orthonormal basis for the Krylov subspace using Arnoldi's process. The input is a non-zero vector $v_1 \ni \|v_1\|_2 = 1$ and a matrix A. Clearly, A is $n \times n$ and v_1 is $n \times 1$. The output is an orthonormal basis for the Krylov Subspace $K_m = \text{span}\{v_1, Av_1, \ldots, A^{m-1}v_1\}$. The matrix whose columns form this basis will be called V_m.

```
function [Hbar,v] =modgs(A,v,m)
[n,n] = size(A);
for j= 1:m,
   vj=v(1:n,j);
   w=A*vj;
   for i=1:j,
        vi=v(1:n,i);
        Hbar(i,j)=vi'*w;
        w=w-Hbar(i,j)*vi
    end
    Hbar(j+1,j)=norm(w,2);
    v=[v,w/Hbar(j+1,j)];
end
```

For example, if the input parameters were:

```
A =

    1    2    3

   -1    4    5

    2    7    4

m = 3

v = [.6,.8,0]',
```

The output would be: $v = \begin{pmatrix} 0.6000 & 0.0235 & -.7997 & -0.5653 \\ 0.8000 & -0.0176 & 0.5997 & 0.5653 \\ 0.0000 & 0.9996 & 0.0294 & 0.6007 \end{pmatrix}$.

Remark: The first three columns of v form the desired orthonormal basis for the Krylov subspace. The eigenvalues of A can be approximated by those of H_m, where $H_m = V_m^T A V_m$. H_m represents the restriction of the linear transformation of A to the subspace K_m with respect to the orthonormal basis

V_m obtained by Arnoldi's process. Let λ_i be the eigenvalues of H_m, $i = 1, 2, \ldots, m$, and let y_i be the corresponding eigenvectors. In other words, $H_m y_i = \lambda_i y_i$, $i = 1, 2, \ldots, m$. As m gets larger, λ_i becomes closer to an eigenvalue of A and $V_m y_i$ gets closer to the corresponding eigenvector of A.

Claim: $v^T(A(V_m y_i) - \lambda_i(V_m y_i)) = 0, \ \forall \ v \in K_m, i = 1, 2, \ldots, m.$

Proof: $v^T(A(V_m y_i) - \lambda_i(V_m y_i))$

$= v^T(A(V_m y_i) - (V_m \lambda_i y_i))$

$= v^T(A(V_m y_i) - (V_m V_m^T A V_m y_i))$ since $V_m^T A V_m y_i = \lambda_i y_i$.

Now note that $V_m^T V_m = I$. This is true because V_m is a set of orthonormal vectors.

Thus, $v^T(A(V_m y_i) - (V_m V_m^T A V_m y_i)) = v^T(A(V_m y_i) - A V_m y_i) = 0$. The result we just proved is called Galerkin condition.

The Full Orthogonalization Method

The full orthogonalization method approximates the solution of the linear system Ax = b. The algorithm starts with an initial residual vector r_0, where $r_0 = b - Ax_0$. Let $v_1 = \frac{r_0}{\|r_0\|_2}$, and $K_m = \text{span}\{v_1, A^2 v_1, A^3 v_1, \ldots\ldots, A^{m-1} v_1\}$.

An approximate solution can be obtained by $x_m = x_0 + z_m$ with $z_m \in K_m$. In the FOM the approximated solution is forced to satisfy the Galerkin condition: $v^T r_m = 0$, for all $v \in K_m$. Let $V_m = \{v_1, v_2, \ldots, v_m\}$ be the orthonormal basis for the above-mentioned Krylov subspace. Thus, $z_m \in K_m$ can be written as a linear combination of $v_1, v_2, \ldots, v_m$, say $z_m = V_m y_m$. Therefore, $x_m = x_0 + V_m y_m$. Since $Ax_0 = b - r_0$, $Ax_m = b - r_m$, and $v_1 = \frac{r_0}{\|r_0\|_2}$, then we have $Ax_m = Ax_0 + AV_m y_m$.

$$A(x_m - x_0) = AV_m y_m$$

$$r_m - r_0 = AV_m y_m$$

$$r_m = r_0 + AV_m y_m$$

$$r_m = \|r_0\|_2 v_1 + AV_m y_m$$

Hence, $V_m^T r_m = V_m^T(\|r_0\|_2 v_1 + AV_m y_m)$

$$= \|r_0\|_2 V_m^T v_1 + V_m^T AV_m y_m$$

$$= \|r_0\|_2 e_1 + H_m y_m,$$

where e_1 is the first column of I_m.

Thus, $y_m = H_m^{-1} \|r_0\|_2 e_1$.

Note that in the above statement $V_m^T v_1$ was replaced by e_1. This can be easily seen by the following argument:

$$V_m^T v_1 = \begin{bmatrix} v_1^T \\ v_2^T \\ v_3^T \\ \cdot \\ \cdot \\ v_m^T \end{bmatrix} \bullet v_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \cdot \\ \cdot \\ 0 \end{bmatrix}$$ (since V_m is a set of orthonormal vectors

The Generalized Minimal Residual Method

The generalized minimal residual method is similar to the FOM method. The only difference between the two methods is that when the GMRES method is used, z_m is chosen in such a way to minimize $\|b - Ax_m\|_2$. It can also be shown that the GMRES method is a projection method with the subspace L as explained earlier equal to AK, where K is the Krylov subspace. The GMRES is used when the coefficient matrix A is nonsymmetric positive definite. If A is symmetric and positive definite, then the Conjugate Gradient method, a projection method, yields the exact solution.

We note that at the end of each iteration of Arnoldi's process we get: $h_{j+1,j}v_{j+1} = Av_j - h_{1j}v_1 - h_{2j}v_2 - \ldots. - h_{jj}v_j$ which is equivalent to $Av_j = v_1h_{1j} + v_2h_{2j} + \ldots + v_jh_{jj} + v_{j+1}h_{j+1,j}$. Let $K_{m+1} = [v_1, v_2, \ldots., v_{m+1}]$ and let V_{m+1} be the matrix whose columns consist of the vectors of K_{m+1}. Then, $AV_m = V_{m+1}\bar{H}_m$,

where $\bar{H}_m$ is an $(m+1)\times m$ matrix whose last row is $[0,0,\ldots,0,h_{m+1,m}]$ and whose other rows are identical to those of H_m. Again we let $x_m = x_0 + z_m$, where z_m is an element of the Krylov subspace chosen in a way to minimize $\|b - Ax_m\|_2$. Let $z_m = V_m y$.

Thus, $\|b - Ax_m\|_2 = \|b - A(x_0 + V_m y)\|_2 = \|r_0 - AV_m y\|_2 = \|\beta v_1 - AV_m y\|_2$.

Since $AV_m = V_{m+1}\bar{H}_m$, then $\|\beta v_1 - AV_m y\|_2 = \left\|V_{m+1}(\beta v_1 - \bar{H}_m y)\right\|_2$. Therefore, minimizing $\|b - Ax_m\|_2$ is equivalent to minimizing $\left\|V_{m+1}(\beta v_1 - \bar{H}_m y)\right\|_2$. However $\|V_{m+1}\|_2$ is a constant. Hence the minimization problem is reduced to minimizing only $\left\|(\beta v_1 - \bar{H}_m y)\right\|_2$. This least squares problem can be solved using QR factorization. See, for example, [1].

Theorem: If $m \geq$ the degree of the minimal polynomial of A, then the exact solution of (I) belongs to $K_m(A,b)$.

The degree of the minimal polynomial of a matrix A is the smallest m that makes $\{I, A, A^2, \ldots, A^m\}$ linearly dependent.

For example, if $A = \begin{pmatrix} 5 & -6 & -6 \\ -1 & 4 & 2 \\ 3 & -6 & -4 \end{pmatrix}$ it can be shown that

$\{I, A, A^2\}$ is the smallest set whose elements are linearly dependent. Hence, the degree of the minimal polynomial of A is 2. In order to find it we would have to solve $A^2 + aA + bI = 0$. It is not hard to show that $a = -3$ and $b = -2$.

Example

Let $A = \begin{pmatrix} 5 & -6 & -6 \\ -1 & 4 & 2 \\ 3 & -6 & -4 \end{pmatrix}$, $b = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$. The eigenvalues of A are 2,2,and 1.The minimal polynomial of A is $q(t) = (t - 2)(t - 1)$ as illustrated above. The exact solution of $Ax = b$ is

$\begin{pmatrix} 14 \\ -3.5 \\ 15 \end{pmatrix}$, and $K_2(A,b) = \left\{ \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \begin{pmatrix} -25 \\ 13 \\ -21 \end{pmatrix} \right\}$

Now $\begin{pmatrix} 14 \\ -3.5 \\ 15 \end{pmatrix} \in K_2(A,b)$, since $\begin{pmatrix} 14 \\ -3.5 \\ 15 \end{pmatrix} = 1.5 \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} - 0.5 \begin{pmatrix} -25 \\ 13 \\ -21 \end{pmatrix}$.

CHAPTER FIVE

CONCLUSION

To conclude this study I will compare and analyze the different numerical techniques discussed in the earlier chapters. The Courtois matrix will be used to compare some of the numerical methods discussed in this study. First, the direct methods such as Gaussian elimination and LDU decomposition can be used to solve relatively small linear systems of equations. However when it comes to solving large sparse matrices like those that arise in solving Markov chains, direct methods display a major disadvantage. Remember that direct methods are based on creating zeros at certain locations of the coefficient matrix. In doing so, unfortunately, non-zero entries are created in other locations that originally contained zeros. This process is called fill-in and it destroys the sparse structure of the original matrix. Consequently, large computer storage would be needed to store such a matrix with lots of non-zero entries. Nevertheless, direct methods have the advantage of enabling us to determine the number of steps required to solve a linear system.

Second, iterative methods do not suffer from having to have large computer storage as direct methods do. The reason for that follows from the fact that iterative methods do not alter the coefficient matrix and hence no fill-ins are created. Unfortunately, when iterative methods are used, we cannot predict the number of iterations required to find a solution with a preset error. There are two factors that govern the convergence rate. The first one is the ratio of the dominant eigenvalue to the subdominant eigenvalue. If this ratio is close to one, the rate of convergence becomes very slow. An example of a matrix that possesses such a behavior is the Courtois matrix. Actually, the Courtois matrix belongs to the class of nearly decomposable matrices usually abbreviated by NCD. If iterative methods were to be applied to a an NCD matrix, then preconditioning techniques would be required to speed up the rate of convergence. The second factor is the initial approximation. In fact, this factor is an advantage when using iterative methods. When an experiment is performed several times, usually the parameters are slightly changed and hence we expect the solutions to be approximately the same. So the calculated solution of a given experiment, for example, can be used as an initial solution to solve the system obtained by a subsequent experiment. Projection methods such as Arnoldi's method and GMRES are efficient to tackle NCD Markov chains. However a storage problem arises when the dimension, m, of the subspace used is large. In this case we use a value of m that would be suitable for the available computer memory. If the resulting approximate solution is not satisfactory, then we use it as an initial approximation. These methods are referred to as iterative Arnoldi's method and iterative GMRES. Now I will present results of some MATLAB's programs to demonstrate some of the facts discussed above. Table 1 shows the eigenvalues of the transpose of the Courtois matrix in the first column. In the second column, we have the eigenvalues of the iteration matrix found when applying the method of Jacobi to the Courtois matrix.

Table 1: Comparison of Eigenvalues of Iteration Matrices

P^T	H_J
1.0	1.0
0.99980000000000	0.99938353640096
0.99849479689134	0.99790895882791
0.75002623150850	0.99579335294684
0.55006663986827	0.58325797025592
0.40003338516447	0.50277001092246
0.30071431880876	0.50277001092246
0.14953537224133	0.41640735625114

Note that in both columns the ratio of the dominant eigenvalue to the subdominant eigenvalue is close to one. Actually, this ratio in the first column is larger than its counterpart in the second column. Thus we expect the Jacobi method to converge faster than the power method. Techniques such as preconditioning can be used to tackle the problem that arises when this ratio is close to one. These techniques can be found in [1].

BIBLIOGRAGHY

1. Stewart W. J. Introduction to the numerical solution of Markov Chains: Princeton University Press. Princeton, New Jersey, 1994.
2. B. Philippe, Y. Saad, and Stewart. Numerical Methods in Markov Chain modeling. Operations Research, 40(6): 1156-1179,1992.
3. Sheldon Axler. Linear Algebra Done Right, 2nd edition: Springer, New York, 1999.
4. Howard M. Taylor, Samuel Karlin. An Introduction to Stochastic Modeling, 3rd edition: Academic Press, San Diego, 1998.