Boundary Value Methods for Transient Solutions of Markovian Queueing Networks

by

MA Ka Chun

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Mathematics

August 2004

Abstract

Abstract of thesis entitled:

Boundary Value Methods for Transient Solutions of Markovian Queueing Networks

Submitted by MA Ka Chun

for the degree of Master of Philosophy in Mathematics

at The Chinese University of Hong Kong in August 2004

In solving the queuing networks in steady states, one has to solve a system of linear equations. For transient states solution, on the other hand, one has to solve a system of ordinary differential equations. The initial value method is one of the classical methods to solve the system. However, a small time step should be used in order for the methods to be stable. This will result in a higher computational cost. In this thesis, the boundary value method is proposed to solve the problem. The boundary value method, which is unconditionally stable even for high-order scheme, will transform the problem into a large linear system. The algebraic multigrid methods are used to reduce the cost of solving the linear system. Numerical results will be given to illustrate the effectiveness of the method.

摘要

香港中文大學碩士論文摘要

論文題目：

馬爾可夫排隊網絡瞬時解的邊界值方法

馬家俊

二零零四年六月

在找出排隊網絡的定態解時，我們需要求解綫性方程組。但對於排隊網絡的瞬時解，我們則要求解常微分方程組。初值方法是找常微分方程數值解的傳統方法，但它的缺點在於它的不穩定性。如果要令初值方法變成穩定，則需要採用一個極小的步長，但這樣會使計算量大大增加。這篇論文提出另一個找出排隊網絡瞬時解的方法，就是利用邊界方值法去求常微分方程組的數值解。邊界值方法是無條件地穩定的數值方法，即使高階的邊界值方法也是一樣。邊界值方法的缺點是它會把常微分方程組變換成一組更大的綫性方程組，但是，透過利用代數多重網格法解這個綫性方程組，計算量就能大大減少，而這篇論文的數值結果亦能顯示這種方法的效率。

ACKNOWLEDGMENTS

I would like to thank my supervisors, Profs. Raymond H. Chan for his guidance in my studies. Besides, I would like to thank Profs. W.K. Ching, H. Sun, and Michael C. Wong, for their help in this project. Thanks also go to Profs. Philippe G. Ciarlet, K.M. Yeung, J. Zou, for their excellent teaching in my graduate courses. Last but not least, I am grateful to my colleagues Mr. Z.J. Bai, C. Hu, C.W. Ho, K.T. Ling, Y.H. Tam, C.Y. Wong, and Y.S. Wong for their many helpful discussions and providing an excellent learning environment in our office.

Contents

Chapter 1

Introduction

Markovian queueing networks are common stochastic models for a number of physical systems such as telecommunication systems [9], manufacturing systems [10] and inventory systems [11]. For long-run system performance analysis, the steady-state probability distribution of the system is required. The steady-state probability distribution can be obtained by solving a large linear system. Direct methods [5, 12, 14] and iterative methods [3, 7, 9, 10] have been developed for this purpose.

However, to analyze the system in a finite horizon, the transient solution of the queueing system is required, and it can be found by solving a system of ordinary differential equations (ODEs). Many classical numerical methods can be applied to the ODE systems. The Initial Value Methods (IVMs) such as the Runge-Kutta method are good explicit methods for their efficiency and easy implementation. But they may require small time step in order to converge. A survey on numerical methods for solving transient solutions of homogeneous irreducible Markov chains can be found in [18].

In this thesis, the Boundary Value Methods (BVMs) [2] are proposed to the

ODE systems. BVMs are implicit stable methods and hence there is no restriction on the size of the time step for the method to converge. However, the disadvantage is that they require solutions of large linear systems, and hence may require longer computational time when compared with the IVMs. Here we propose to use the algebraic multigrid (AMG) method to solve the resulting linear systems from BVMs.

AMG methods have been developed for more than two decades [20] and have been applied to many applications such as solving partial differential equations [1] and imaging problems [16]. They also have been used successfully for finding the steady-state probability distributions of queueing networks by using an appropriate coarse grid approximation, see [7]. In this paper, we use AMG methods for queues in transient states. The ODEs are first discretized by BVMs and the resulting linear systems are solved by AMG methods. For overflow queueing networks, we will see that the resulting method is much more efficient than IVMs, especially when the systems are ill-conditioned. We will illustrate the effectiveness of our method through 2-queue overflow networks. A comparison with other iterative methods will also be given.

The thesis is organized as follows. In Chapter 2, we present the overflow queues. IVMs and BVMs are introduced in Chapter 3 while the linear system solvers are given in Chapter 4. In Chapter 5, numerical examples are given to demonstrate the efficiency of our method.

Chapter 2

Queueing Networks

For continuous-time Markovian queueing networks, the transient probability distribution can be found by solving Kolmogorov's backward equations [8, 21]. We introduce the equations here.

2.1 One-queue Networks

We will focus on the birth and death processes first. Suppose that there are n states,

$$\{0, 1, 2, \ldots, n-1\}.$$

We denote $q_i(t)$ be the probability that it is in state i at time t provided that it is at state 0 at time 0. We define

$$\mathbf{q}(t) = \begin{bmatrix} q_0(t) \\ q_1(t) \\ \vdots \\ q_{n-1}(t) \end{bmatrix}$$

then the Kolmogorov backward equation is

$$\frac{d\mathbf{q}(t)}{dt} = -G\mathbf{q}(t), \tag{2.1}$$

where

$$G = \begin{bmatrix} \lambda & -\mu & & & & & 0 \\ -\lambda & \lambda+\mu & -2\mu & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & -\lambda & \lambda+s\mu & -s\mu & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & -\lambda & \lambda+s\mu & -s\mu \\ 0 & & & & & -\lambda & s\mu \end{bmatrix}, \tag{2.2}$$

s is the number of servers, λ and μ are the mean arrival rate and service rate respectively, and clearly we have the initial condition $\mathbf{q}(0) = (1, 0, \ldots, 0)^T \in \mathbb{R}^n$. This one-queue network is called a Markovian $M/M/s/(n-s-1)$ queue. The true solution for this network can be written as

$$\mathbf{q}(t) = e^{-Gt}\mathbf{q}(0) = \left(I + \sum_{j=1}^{\infty} \frac{(-G)^j t^j}{j!} \right) \mathbf{q}(0).$$

We can, on the other hand, find the solution in another way. If we define

$$S = a \begin{bmatrix} 1 & & & & & & \\ & \left(\frac{\lambda}{\mu}\right)^{\frac{1}{2}} & & & & 0 & \\ & & \left(\frac{\lambda}{\mu}\right)^{\frac{1}{2}}\left(\frac{\lambda}{2\mu}\right)^{\frac{1}{2}} & & & & \\ & & & \ddots & & & \\ & & & & \left(\frac{\lambda}{\mu}\right)^{\frac{1}{2}} \cdots \left(\frac{\lambda}{s\mu}\right)^{\frac{1}{2}} & & \\ & 0 & & & & \ddots & \\ & & & & & & \prod_{j=1}^{n-1}\left(\frac{\lambda}{\min(j,s)\mu}\right)^{\frac{1}{2}} \end{bmatrix},$$

where

$$a = \frac{1}{1 + \sum_{i=2}^{n} \prod_{j=1}^{i-1} \left(\frac{\lambda}{\min(j,s)\mu}\right)^{\frac{1}{2}}},$$

then by direct calculation, we will have

$$(S^{-1}GS)^T = S^{-1}GS,$$

i.e., $S^{-1}GS$ is symmetric. Thus there exists an orthogonal matrix Q with a diagonal matrix

$$\Gamma = \begin{bmatrix} \Gamma_1 & & & 0 \\ & \Gamma_2 & & \\ & & \ddots & \\ 0 & & & \Gamma_n \end{bmatrix},$$

such that

$$Q^T S^{-1} G S Q = \Gamma.$$

Let $\tilde{\mathbf{q}}(t)$ be such that $\mathbf{q}(t) = SQ\tilde{\mathbf{q}}(t)$. Then (2.1) will become

$$\begin{aligned} \frac{d\mathbf{q}(t)}{dt} &= -G\mathbf{q}(t) \\ \frac{dSQ\tilde{\mathbf{q}}(t)}{dt} &= -GSQ\tilde{\mathbf{q}}(t) \\ \frac{d\tilde{\mathbf{q}}(t)}{dt} &= -\Gamma\tilde{\mathbf{q}}(t). \end{aligned}$$

By using the above auxiliary equation, the solution of (2.1) is

$$\begin{aligned} \mathbf{q}(t) = SQ\tilde{\mathbf{q}}(t) &= SQe^{-\Gamma t}\tilde{\mathbf{q}}(0) \\ &= SQ \begin{bmatrix} e^{-\Gamma_1 t} & & & 0 \\ & e^{-\Gamma_2 t} & & \\ & & \ddots & \\ 0 & & & e^{-\Gamma_n t} \end{bmatrix} Q^T S^{-1}\mathbf{q}(0). \end{aligned}$$

The one-queue solution can be found exactly by using the above formula. This kind of solution is the transient solution of the network. On the other hand, one may consider the behavior when the network reaches an equilibrium after a sufficiently long time. This solution is called the steady-state solution. The

steady-state solution can be found by imposing

$$\frac{d\mathbf{q}(t)}{dt} = 0,$$

i.e., finding a vector $\mathbf{q}$ such that

$$G\mathbf{q} = 0.$$

It is easy to see that the matrix G has a left eigenvector $(1, 1, \ldots, 1) \in \mathbb{R}^n$ with eigenvalue 0. So one more constraint should be added to make $\mathbf{q}$ a probability distribution, i.e., $\mathbf{q}$ should be non-negative with $\|\mathbf{q}\|_1 = 1$. The focus of this thesis will be on the transient solution, and more results on the steady-state solution can be found in [3].

Now that the results for one-queue networks have been reviewed. The following section will be about the two-queue networks.

2.2 Two-queue Free Networks

We begin our discussion of a simple two-queue free network first. In this network, there are no interactions between the two Markovian $M/M/s_i/(n_i - s_i - 1)$ queues, $i = 1, 2$. Here s_i and $(n_i - s_i - 1)$ denote the number of parallel servers and the number of queueing spaces in Queue i respectively. In state (i, j), we mean that there are i customers in Queue 1 and j customers in Queue 2. For simplicity of notations, let $p_{i,j}(t)$ be the probability that the network is in state (i, j) at time t provided at time 0 it is in state $(0, 0)$. In general, the initial state can be arbitrary. If we let

$$\begin{aligned}\mathbf{p}(t) =&(p_{0,0}(t), \ldots, , p_{0,n_2-1}(t), p_{1,0}(t), \ldots, p_{1,n_2-1}(t),\\ &\ldots, p_{n_1-1,0}(t), \ldots, p_{n_1-1,n_2-1}(t))^T,\end{aligned}$$

then the Kolmogorov backward equations can be written as

$$\frac{d\mathbf{p}(t)}{dt} = -(G_1 \otimes I_{n_2} + I_{n_1} \otimes G_2)\mathbf{p}(t), \tag{2.3}$$

where

$$G_i = \begin{bmatrix} \lambda_i & -\mu_i & & & & & 0 \\ -\lambda_i & \lambda_i + \mu_i & -2\mu_i & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & -\lambda_i & \lambda_i + s_i\mu_i & -s_i\mu_i & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & -\lambda_i & \lambda_i + s_i\mu_i & -s_i\mu_i \\ 0 & & & & & -\lambda_i & s_i\mu_i \end{bmatrix}, \tag{2.4}$$

and λ_i and μ_i are the mean arrival rate and the mean service rate of the servers in Queue i. Since there are no interactions between the queues, the transient solution can be obtained in tensor product forms from the solutions of individual queues, i.e., if $\mathbf{q}_i(t)$ is the transient solution for Queue i, i =1, 2, then $(\mathbf{q}_1(t) \otimes \mathbf{q}_2(t))$ is the transient solution for (2.3). In fact,

$$\begin{aligned} \frac{d}{dt}(\mathbf{q}_1(t) \otimes \mathbf{q}_2(t)) &= \frac{d\mathbf{q}_1(t)}{dt} \otimes \mathbf{q}_2(t) + \mathbf{q}_1(t) \otimes \frac{d\mathbf{q}_2(t)}{dt} \\ &= -G_1\mathbf{q}_1(t) \otimes \mathbf{q}_2(t) - \mathbf{q}_1(t) \otimes G_2\mathbf{q}_2(t) \\ &= -(G_1 \otimes I_{n_2} + I_{n_1} \otimes G_2)(\mathbf{q}_1(t) \otimes \mathbf{q}_2(t)). \end{aligned}$$

The two-queue free network is not very interesting because the solution can be found explicitly. But when some interactions are allowed between the two networks, the problem will get more complicated as we will see in the next section.

2.3 Two-queue Overflow Networks

In this section we present two-queue overflow networks. Unlike two-queue free networks, they allow overflow from one queue to another. Here we consider the

following overflow discipline:

1. When the first queue is full, customers arriving at Queue 1 are allowed to overflow to Queue 2 if it is not yet full.

2. Overflow from Queue 2 to Queue 1 is not allowed.

Then the transient solution $\mathbf{p}(t)$ satisfies

$$\frac{d\mathbf{p}(t)}{dt} = -(G_1 \otimes I_{n_2} + I_{n_1} \otimes G_2 + \mathbf{e}_{n_1}\mathbf{e}_{n_1}^T \otimes R_1)\mathbf{p}(t), \tag{2.5}$$

where $\mathbf{e}_{n_1}^T = (0, \ldots, 0, 1) \in \mathbb{R}^{n_1}$ and

$$R_1 = \lambda_1 \cdot \begin{bmatrix} 1 & & & & 0 \\ -1 & 1 & & & \\ & \ddots & \ddots & & \\ & & -1 & 1 & \\ 0 & & & -1 & 0 \end{bmatrix}.$$

For two-queue overflow networks, there are no product form solutions, and thus numerical methods must be used to find the solutions. One can easily extend these ideas to obtain the backward equations for more general q-queue overflow networks, see [4].

2.4 Networks with Batch Arrivals

Unlike birth and death processes, this network allow batch arrivals. A one-queue network with batch arrivals can be described as follows. Let $\tilde{\lambda}_k$ be the batch arrival rate for batches with size k, $k \geq 1$, and define

$$\tilde{\lambda} = \sum_{j=1}^{\infty} \tilde{\lambda}_j.$$

If the arrival batch size is larger than the places available, only part of the batch can arrive the system. With the same definition as in §2.1, the probability distribution $\mathbf{q}(t)$ satisfies

$$\frac{d\mathbf{q}(t)}{dt} = -\tilde{G}\mathbf{q}(t),$$

where

$$\tilde{G} = \begin{bmatrix} \tilde{\lambda} & -\mu & 0 & 0 & 0 & \cdots & 0 \\ -\tilde{\lambda}_1 & \tilde{\lambda}+\mu & -2\mu & 0 & 0 & \cdots & 0 \\ -\tilde{\lambda}_2 & -\tilde{\lambda}_1 & \tilde{\lambda}+2\mu & \ddots & \ddots & \ddots & \vdots \\ \vdots & -\tilde{\lambda}_2 & \ddots & \ddots & -s\mu & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & \tilde{\lambda}+s\mu & \ddots & 0 \\ -\tilde{\lambda}_{n-2} & -\tilde{\lambda}_{n-3} & \cdots & -\tilde{\lambda}_2 & -\tilde{\lambda}_1 & \tilde{\lambda}+s\mu & -s\mu \\ -\sum_{j=n-1}^{\infty}\tilde{\lambda}_j & -\sum_{j=n-2}^{\infty}\tilde{\lambda}_j & \cdots & \cdots & \cdots & -\sum_{j=1}^{\infty}\tilde{\lambda}_j & s\mu \end{bmatrix}. \quad (2.6)$$

In the next chapter, some of the numerical methods for solving the ODE will be discussed.

Chapter 3

ODE Solvers

In this chapter, we will introduce the classical initial value methods and the boundary value methods. The convergence and stability of the methods will also be discussed.

3.1 The Initial Value Methods

3.1.1 The Linear System of Ordinary Differential Equations

For general Kolmogorov's backward equations, it is in the form:

$$\frac{d\mathbf{p}(t)}{dt} = -H\mathbf{p}(t), \tag{3.1}$$

where H depends on the queueing networks.

Before describing the IVMs, we define the notation for the discretization. Suppose we want to find $\mathbf{p}(T)$ for some final time $T < \infty$. Then we divide the time horizon into N steps, with step size $h = T/N$. Denote $\mathbf{p}_k = \mathbf{p}(kh)$,

$0 \leq k \leq N$, the probability distribution that we want to find. With these notations, some of the classical IVMs will be reviewed.

3.1.2 Euler's Method

The forward Euler method can be derived by using the following approximation,

$$\frac{d\mathbf{p}(kh)}{dt} \approx \frac{\mathbf{p}(kh+h) - \mathbf{p}(kh)}{h},$$

for $k = 0, 1, \ldots, N-1$. The forward Euler formula then become

$$\mathbf{p}_{k+1} = (I - hH)\mathbf{p}_k. \tag{3.2}$$

The forward Euler method is an explicit method, and it requires only matrix-vector multiplication for finding the approximation for next time step if the previous one is known. By Taylor's expansion,

$$\mathbf{p}(kh+h) = \mathbf{p}(kh) + h\frac{d\mathbf{p}(t)}{dt} + O(h^2).$$

As a result, the approximation in the Euler method is of order $O(h)$.

3.1.3 Runge-Kutta Methods

Another most powerful IVM for solving a general ODE is the Runge-Kutta method. We will consider the Runge-Kutta method of order 4 (RK4). RK4 can be used to solve the ODEs of the following type:

$$\frac{d\mathbf{p}(t)}{dt} = f(t, \mathbf{p}(t)),$$

where f is an arbitrary function. The well-known RK4 formula is as follows,

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4),$$

where

$$\begin{aligned}
K_1 &= f(kh, \mathbf{p}_k), \\
K_2 &= f(kh + \frac{h}{2}, \mathbf{p}_k + h\frac{K_1}{2}), \\
K_3 &= f(kh + \frac{h}{2}, \mathbf{p}_k + h\frac{K_2}{2}), \\
K_4 &= f(kh + h, \mathbf{p}_k + hK_3).
\end{aligned}$$

In our case,

$$f(t, \mathbf{p}(t)) = -H\mathbf{p}(t),$$

therefore

$$\begin{aligned}
K_1 &= f(kh, \mathbf{p}_k) \\
&= -H\mathbf{p}_k, \\
K_2 &= f(kh + \frac{h}{2}, \mathbf{p}_k + h\frac{K_1}{2}) \\
&= f(kh + \frac{h}{2}, \mathbf{p}_k - \frac{h}{2}H\mathbf{p}_k) \\
&= -H(\mathbf{p}_k - \frac{h}{2}H\mathbf{p}_k) \\
&= (-H + \frac{h}{2}H^2)\mathbf{p}_k, \\
K_3 &= f(kh + \frac{h}{2}, \mathbf{p}_k + \frac{h}{2}(-H + \frac{h}{2}H^2)\mathbf{p}_k) \\
&= -H(\mathbf{p}_k + \frac{h}{2}(-H + \frac{h}{2}H^2)\mathbf{p}_k) \\
&= (-H + \frac{h}{2}H^2 - \frac{h^2}{2^2}H^3)\mathbf{p}_k, \\
K_4 &= f(kh + h, \mathbf{p}_k + h(-H + \frac{h}{2}H^2 - \frac{h^2}{2^2}H^3)\mathbf{p}_k) \\
&= -H(\mathbf{p}_k + h(-H + \frac{h}{2}H^2 - \frac{h^2}{2^2}H^3)\mathbf{p}_k) \\
&= (-H + hH^2 - \frac{h^2}{2}H^3 + \frac{h^3}{2^2}H^4)\mathbf{p}_k.
\end{aligned}$$

As a result, RK4 can be simplified as follows:

$$\begin{aligned}
\mathbf{p}_{k+1} =& \mathbf{p}_k + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \\
=& \mathbf{p}_k + \frac{h}{6}\left(-H\mathbf{p}_k + 2(-H + \frac{h}{2}H^2)\mathbf{p}_k \right. \\
& + 2(-H + \frac{h}{2}H^2 - \frac{h^2}{2^2}H^3)\mathbf{p}_k \\
& \left. + (-H + hH^2 - \frac{h^2}{2}H^3 + \frac{h^3}{2^2}H^4)\mathbf{p}_k\right) \\
=& \mathbf{p}_k + \frac{h}{6}\left(-6H + 3hH^2 - h^2H^3 + \frac{h^3}{2^2}H^4\right)\mathbf{p}_k, \\
\mathbf{p}_{k+1} =& (I - hH + \frac{1}{2}h^2H^2 - \frac{1}{6}h^3H^3 + \frac{1}{24}h^4H^4)\mathbf{p}_k, \qquad (3.3)
\end{aligned}$$

for $k = 0, 1, \ldots, N-1$. Like the Euler method, RK4 requires only matrix-vector multiplications. By Taylor's expansion,

$$\mathbf{p}(kh+h) = \mathbf{p}(kh) + h\frac{d\mathbf{p}(kh)}{dt} + \frac{h^2}{2}\frac{d^2\mathbf{p}(kh)}{dt^2} + \frac{h^3}{6}\frac{d^3\mathbf{p}(kh)}{dt^3} + \frac{h^4}{24}\frac{d^4\mathbf{p}(kh)}{dt^4} + O(h^5).$$

So the approximation is of order $O(h^4)$.

3.1.4 The Stability of the IVMs

The stability of the IVMs is a good criterion for choosing the most suitable numerical method. In general, if an IVM can be expressed as:

$$\mathbf{p}_{k+1} = T\mathbf{p}_k,$$

then the method is stable if $\|T\| < 1$. So for the Euler method, by (3.2), it is stable if

$$\|I - hH\| < 1.$$

For RK4, it is stable if

$$\|I - hH + \frac{1}{2}h^2H^2 - \frac{1}{6}h^3H^3 + \frac{1}{24}h^4H^4\| < 1,$$

by (3.3).

For the Euler method, using the triangle inequality for matrix norm,

$$\|hH\| - \|I\| \leq \|I - hH\|.$$

We can then derive a necessary condition for the Euler method to be stable,

$$\begin{aligned} \|hH\| &\leq \|I - hH\| + \|I\| \\ h\|H\| &< 2 \\ h &< \frac{2}{\|H\|}. \end{aligned}$$

So if the norm of H increases, the time step size h should decrease accordingly in order to keep the method stable. This result is also true for RK4.

3.1.5 Applications in Queueing Networks

For the overflow network in (2.5), we can take

$$H = G_1 \otimes I_{n_2} + I_{n_1} \otimes G_2 + \mathbf{e}_{n_1}\mathbf{e}_{n_1}^T \otimes R_1. \tag{3.4}$$

The IVMs are efficient for finding the transient solution, but we note that for some queueing parameters, $\|H\|$ may increase as the sizes of the queues increase. In these cases, h has to be small in order that RK4 converges, and this will render the IVMs to be less efficient.

For the batch arrival network, take $H = \tilde{G}$ where $\tilde{G}$ is defined in (2.6).

3.2 The Boundary Value Methods

The BVMs are stable implicit methods which can be used especially if the problem is ill-conditioned. The Generalized Backward Differentiation Formulae (GBDF) which is an important class of BVMs will be presented.

3.2.1 The Generalized Backward Differentiation Formulae

We define $\mathbf{f}_k = f(t_k, \mathbf{p}_k)$, for $k = 0, 1, \ldots, N$. The GBDF of order m can be defined as

$$\sum_{i=0}^{m} \alpha_i \mathbf{p}_{k+i} = h\mathbf{f}_{n+\nu},$$

for $k = 0, 1, \ldots, N - m$, where

$$\nu = \begin{cases} \frac{k+2}{2} \text{ if } k \text{ is even,} \\ \frac{k+1}{2} \text{ if } k \text{ is odd,} \end{cases}$$

and the coefficients α_i are chosen to make the approximation to be of $O(h^m)$. We require m more equations to find all $\mathbf{p}_k$. Suppose we add μ initial conditions and $m - \mu$ final conditions. The discrete problem for (3.1) will then become

$$(A \otimes I + hB \otimes H) \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_N \end{bmatrix} = \mathbf{e}_1 \otimes \mathbf{p}_0, \tag{3.5}$$

where $\mathbf{e}_1 = (1, 0, \ldots, 0)^T \in \mathbb{R}^{N+1}$, A and B are $(N+1)$-by-$(N+1)$ matrices:

$$A = \begin{bmatrix}
1 & \cdots & 0 & & & & \\
\alpha_0^{(1)} & \cdots & \alpha_m^{(1)} & & & & \\
\vdots & \vdots & \vdots & & & 0 & \\
\alpha_0^{(\mu)} & \cdots & \alpha_m^{(\mu)} & & & & \\
\alpha_0 & \cdots & \alpha_m & & & & \\
 & \alpha_0 & \cdots & \alpha_m & & & \\
 & & \ddots & \ddots & \ddots & & \\
 & & & \ddots & \ddots & \ddots & \\
 & & & & \alpha_0 & \cdots & \alpha_m \\
 & & 0 & & \alpha_0^{(N-m+\mu+1)} & \cdots & \alpha_m^{(N-m+\mu+1)} \\
 & & & & \vdots & \vdots & \vdots \\
 & & & & \alpha_0^{(N)} & \cdots & \alpha_m^{(N)}
\end{bmatrix},$$

and

$$B = \begin{bmatrix}
0 & \cdots & 0 & & & & \\
\beta_0^{(1)} & \cdots & \beta_m^{(1)} & & & & \\
\vdots & \vdots & \vdots & & & & \\
\beta_0^{(\mu)} & \cdots & \beta_m^{(\mu)} & & & 0 & \\
\beta_0 & \cdots & \beta_m & & & & \\
 & \beta_0 & \cdots & \beta_m & & & \\
 & & \ddots & \ddots & \ddots & & \\
 & & & \ddots & \ddots & \ddots & \\
 & & & & \beta_0 & \cdots & \beta_m \\
 & & 0 & & \beta_0^{(N-m+\mu+1)} & \cdots & \beta_m^{(N-m+\mu+1)} \\
 & & & & \vdots & \vdots & \vdots \\
 & & & & \beta_0^{(N)} & \cdots & \beta_m^{(N)}
\end{bmatrix}.$$

Now define $\tilde{A}$ and $\tilde{B}$ be two N-by-N matrices:

$$\tilde{A} = \begin{bmatrix}
\alpha_1^{(1)} & \cdots & \alpha_m^{(1)} & & & & \\
\vdots & \vdots & \vdots & & & 0 & \\
\alpha_1^{(\mu)} & \cdots & \alpha_m^{(\mu)} & & & & \\
\alpha_0 & \cdots & \alpha_m & & & & \\
 & \alpha_0 & \cdots & \alpha_m & & & \\
 & & \ddots & \ddots & \ddots & & \\
 & & & \ddots & \ddots & \ddots & \\
 & & & & \alpha_0 & \cdots & \alpha_m \\
 & & 0 & & \alpha_0^{(N-m+\mu+1)} & \cdots & \alpha_m^{(N-m+\mu+1)} \\
 & & & & \vdots & \vdots & \vdots \\
 & & & & \alpha_0^{(N)} & \cdots & \alpha_m^{(N)}
\end{bmatrix},$$

and

$$\tilde{B} = \begin{bmatrix}
\beta_1^{(1)} & \cdots & \beta_m^{(1)} & & & & \\
\vdots & \vdots & \vdots & & & & \\
\beta_1^{(\mu)} & \cdots & \beta_m^{(\mu)} & & & 0 & \\
\beta_0 & \cdots & \beta_m & & & & \\
 & \beta_0 & \cdots & \beta_m & & & \\
 & & \ddots & \ddots & \ddots & & \\
 & & & \ddots & \ddots & \ddots & \\
 & & & & \beta_0 & \cdots & \beta_m \\
 & & 0 & & \beta_0^{(N-m+\mu+1)} & \cdots & \beta_m^{(N-m+\mu+1)} \\
 & & & & \vdots & \vdots & \vdots \\
 & & & & \beta_0^{(N)} & \cdots & \beta_m^{(N)}
\end{bmatrix}.$$

Then (3.5) can be written as

$$\left(\begin{bmatrix} I & 0 & \cdots & 0 \\ \alpha_0^{(1)} I & & & \\ \vdots & & \tilde{A} \otimes I & \\ \alpha_0^{(\mu)} I & & & \\ \alpha_0 I & & & \end{bmatrix} + \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \beta_0^{(1)} hH & & & \\ \vdots & & \tilde{B} \otimes hH & \\ \beta_0^{(\mu)} hH & & & \\ \beta_0 hH & & & \end{bmatrix} \right) \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_N \end{bmatrix} = \begin{bmatrix} \mathbf{p}_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

By eliminating the redundant equations, we get

$$\begin{bmatrix} \alpha_0^{(1)} \mathbf{p}_0 \\ \vdots \\ \alpha_0^{(\mu)} \mathbf{p}_0 \\ \alpha_0 \mathbf{p}_0 \end{bmatrix} + \begin{bmatrix} \beta_0^{(1)} hH\mathbf{p}_0 \\ \vdots \\ \beta_0^{(\mu)} hH\mathbf{p}_0 \\ \beta_0 hH\mathbf{p}_0 \end{bmatrix} + (\tilde{A} \otimes I + h\tilde{B} \otimes H) \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_N \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

or

$$(\tilde{A} \otimes I + h\tilde{B} \otimes H) \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_N \end{bmatrix} = - \begin{bmatrix} \alpha_0^{(1)} \\ \vdots \\ \alpha_0^{(\mu)} \\ \alpha_0 \end{bmatrix} \otimes \mathbf{p}_0 - h \begin{bmatrix} \beta_0^{(1)} \\ \vdots \\ \beta_0^{(\mu)} \\ \beta_0 \end{bmatrix} \otimes H\mathbf{p}_0.$$

3.2.2 An example

An example of BVMs is the third order generalized backward differentiation formulae (GBDF3):

$$\frac{1}{6}(2\mathbf{p}_{k+1} + 3\mathbf{p}_k - 6\mathbf{p}_{k-1} + \mathbf{p}_{k-2}) = h\mathbf{f}_k, \quad k = 2, 3, \ldots, N-1,$$

$$\frac{1}{6}(-\mathbf{p}_3 + 6\mathbf{p}_2 - 3\mathbf{p}_1 - 2\mathbf{p}_0) = h\mathbf{f}_1,$$

$$\frac{1}{6}(11\mathbf{p}_N - 18\mathbf{p}_{N-1} + 9\mathbf{p}_{N-2} - 2\mathbf{p}_{N-3}) = h\mathbf{f}_N,$$

see [2, p. 132]. In matrix form, it is

$$[A \otimes I + hI_N \otimes H] \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_N \end{bmatrix} = -\mathbf{a} \otimes \mathbf{p}_0, \tag{3.6}$$

where $\mathbf{a}^T = (-\frac{1}{3}, \frac{1}{6}, 0, \ldots, 0) \in \mathbf{R}^N$, and A is the N-by-N matrix

$$A = \begin{bmatrix} -\frac{1}{2} & 1 & -\frac{1}{6} & & & & 0 \\ -1 & \frac{1}{2} & \frac{1}{3} & & & & \\ \frac{1}{6} & -1 & \frac{1}{2} & \frac{1}{3} & & & \\ & \frac{1}{6} & -1 & \frac{1}{2} & \frac{1}{3} & & \\ & & \ddots & \ddots & \ddots & \ddots & \\ & & & \frac{1}{6} & -1 & \frac{1}{2} & \frac{1}{3} \\ 0 & & & -\frac{1}{3} & \frac{3}{2} & -3 & \frac{11}{6} \end{bmatrix}.$$

The GBDF3 is stable in the sense that a very large time step can be used, while the drawback is that we need to solve the big linear system (3.6) which has size N times the size of H. There are many alternative ways to solve (3.6). In the next chapter we will review some of them.

Chapter 4

The Linear Equation Solver

There are many methods to solve a linear system. They include direct methods and indirect methods. Some of the indirect methods will be reviewed in this chapter and the application of them on (3.6) will then be discussed.

4.1 Iterative Methods

Suppose we want to solve the following n linear equations in matrix form,

$$W\mathbf{x} = \mathbf{b}. \tag{4.1}$$

Instead of finding the solution directly, one can find the solution by an iterative formula of the form

$$\mathbf{x}^{(k+1)} = T\mathbf{x}^{(k)} + \mathbf{c},$$

where T is called the iterative matrix. Therefore, given an initial guess $\mathbf{x}_0$, we can find successively $\mathbf{x}_k$, $k \geq 1$. Define $\mathbf{r}_k$ to be the residual vector after k iterations, i.e.,

$$\mathbf{r}_k = \mathbf{b} - W\mathbf{x}_k,$$

for each $k \geq 0$. When we use the iterative method, we may stop the iteration if

$$\frac{\|\mathbf{r}_k\|}{\|\mathbf{r}_0\|} < \epsilon,$$

for all some $k < K$, where ϵ is called the tolerance. In this case we say that the iterative method converges in k iterations at the tolerance level ϵ.

If eventually

$$\frac{\|\mathbf{r}_k\|}{\|\mathbf{r}_0\|} \geq \epsilon,$$

for all $k \leq K$, then we say that the iterative method does not converge at the tolerance level ϵ.

After these basic notations, two important iterative methods will be introduced and more advanced methods based on them will be discussed afterward.

4.1.1 The Jacobi method

The matrix $W = (w_{ij})$ can be decomposed as

$$W = D - L - U,$$

where D is a diagonal matrix, L a strictly lower triangular matrix, and U a strictly upper triangular matrix, i.e.,

$$D = \begin{bmatrix} w_{11} & & & \\ & w_{22} & & \\ & & \ddots & \\ & & & w_{nn} \end{bmatrix},$$

$$L = -\begin{bmatrix} 0 & & & \\ w_{21} & 0 & & \\ \vdots & \ddots & \ddots & \\ w_{n1} & \cdots & w_{n,n-1} & 0 \end{bmatrix},$$

$$U = -\begin{bmatrix} 0 & w_{12} & \cdots & w_{1n} \\ & \ddots & \ddots & \vdots \\ & & 0 & w_{n-1,n} \\ & & & 0 \end{bmatrix}.$$

Then (4.1) can be written as

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

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

The Jacobi method is then defined as

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

Then for each $i = 1, 2, \ldots, n$,

$$x_i^{(k+1)} = \frac{1}{w_{ii}} \left(-\sum_{\substack{j=1 \\ j \neq i}}^{n} w_{ij} x_j^{(k)} + b_i \right),$$

where $\mathbf{x}^{(k+1)} = (x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)})$.

4.1.2 The Gauss-Seidel Method

Using the same notation in §4.1.1, the Gauss-Seidel method can be defined as follows. From (4.1),

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

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

the Gauss-Seidel method is then defined as

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

or for each $i = 1, 2, \ldots, n$,

$$x_i^{(k+1)} = \frac{1}{w_{ii}} \left(- \sum_{j=1}^{i-1} w_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} w_{ij} x_j^{(k)} + b_i \right).$$

4.1.3 Other Iterative Methods

Besides the classical iterative methods like the Jacobi and Gauss-Seidel methods, there are many others like the conjugate gradient method and the generalized minimum residual (GMRES) method. More details of these methods can be found in [17].

4.1.4 Preconditioning

The principle of preconditioning is that we want to find a suitable matrix to decrease the number of iterations required to achieve the desired tolerance level. Suppose we find a matrix $\tilde{W}$ such that $\tilde{W}^{-1}W$ is close to identity. Then instead of solving (4.1), we solve

$$\tilde{W}^{-1}W\mathbf{x} = \tilde{W}^{-1}\mathbf{b}.$$

Here $\tilde{W}$ is called a preconditioner. If it can be found properly, the iterative method can become swifter.

For example, T. Chan's preconditioner $c(G)$ for G where G is defined in §2.1 is

$$c(G) = \begin{bmatrix} \gamma_0 & -\gamma_1 & & & -\lambda \\ -\lambda & \gamma_0 & -\gamma_1 & & \\ & \ddots & \ddots & \ddots & \\ & & -\lambda & \gamma_0 & -\gamma_1 \\ -\gamma_1 & & & -\lambda & \gamma_0 \end{bmatrix},$$

where

$$\gamma_0 = \frac{1}{n} \left((n-1)\lambda + \sum_{j=1}^{n-1} \mu \min\{j, s\} \right),$$

and

$$\gamma_1 = \frac{1}{n-1} \sum_{j=1}^{n-1} \mu \min\{j, s\}.$$

See [6] for more details. The preconditioning method is good for some well-conditioned problem. However, if the problem is ill-conditioned, some other methods are required.

4.2 The Multigrid Method

The multigrid method has been used for solving linear equations discretized from differential equations. The idea of the multigrid method comes from iterative refinement and filtering.

4.2.1 Iterative Refinement

The linear system (4.1) can be solved by iterative refinement as follows. If $\mathbf{x}_0$ is an initial approximation of (4.1). Then for $k \geq 0$, define

$$\mathbf{r}_k = \mathbf{b} - W\mathbf{x}_k.$$

We find $\mathbf{e}_k$ such that

$$W\mathbf{e}_k = \mathbf{r}_k,$$

then $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{e}_k$. This procedure is called iterative refinement.

4.2.2 Restriction and Prolongation

The idea of restriction and prolongation can be explained most easily through solving a one-dimensional Poisson's equation on the interval $[0, 1]$ with Dirichlet homogenous boundary conditions by the finite difference method. Suppose that

the step size is $\frac{1}{2^m}$, where m is an integer, and x_k is the approximation at $\frac{k}{2^m}$, $k = 0, 1, \ldots, 2^m$. If we want to pass this approximation to an approximation $\tilde{x}_k$, $k = 0, 1, \ldots, 2^{m-1}$, at a coarser grid with step size $\frac{1}{2^{m-1}}$, we can do it as follows,

$$\begin{bmatrix} 0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{2^m-3} \\ x_{2^m-2} \\ x_{2^m-1} \\ 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 \\ \tilde{x}_1 \\ \vdots \\ \tilde{x}_{2^{m-1}-1} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{4}(x_1 + 2x_2 + x_3) \\ \vdots \\ \frac{1}{4}(x_{2^m-3} + 2x_{2^m-2} + x_{2^m-1}) \\ 0 \end{bmatrix},$$

or in matrix form,

$$\begin{bmatrix} \tilde{x}_1 \\ \vdots \\ \tilde{x}_{2^{m-1}-1} \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 & & & & & & \\ & & 1 & 2 & 1 & & & & \\ & & & & \cdots & \cdots & \cdots & & \\ & & & & & & 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{2^m-3} \\ x_{2^m-2} \\ x_{2^m-1} \end{bmatrix}.$$

In fact, the one-dimensional restriction operator from grid size $\frac{1}{2^{m+1}}$ to $\frac{1}{2^m}$ is defined as

$$I_{m+1}^m = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 & & & & & & 0 \\ & & 1 & 2 & 1 & & & & \\ & & & & \cdots & \cdots & \cdots & & \\ 0 & & & & & & 1 & 2 & 1 \end{bmatrix},$$

where its size is $(2^m - 1)$-by-$(2^{m+1} - 1)$.

On the other hand, if the approximation $\tilde{x}_k$, $k = 0, 1, \ldots, 2^{m-1}$, at a coarser grid with step size $\dfrac{1}{2^{m-1}}$ being given, and we would like to pass it to an approximation x_k, $k = 0, 1, \ldots, 2^m$, in a finer grid with step size $\dfrac{1}{2^m}$, we can do it as follows,

$$\begin{bmatrix} 0 \\ \\ \tilde{x}_1 \\ \\ \vdots \\ \\ \tilde{x}_{2^{m-1}-1} \\ \\ 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{2^m-3} \\ x_{2^m-2} \\ x_{2^m-1} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{2}(0+\tilde{x}_1) \\ \tilde{x}_1 \\ \frac{1}{2}(\tilde{x}_1+\tilde{x}_2) \\ \vdots \\ \frac{1}{2}(\tilde{x}_{2^{m-1}-2}+\tilde{x}_{2^{m-1}-1}) \\ \tilde{x}_{2^{m-1}-1} \\ \frac{1}{2}(\tilde{x}_{2^{m-1}-1}+0) \\ 0 \end{bmatrix},$$

or in matrix form,

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{2^m-3} \\ x_{2^m-2} \\ x_{2^m-1} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & & & & \\ 2 & & & & \\ 1 & 1 & & & \\ & 2 & & & \\ & 1 & \vdots & & \\ & & \vdots & & \\ & & \vdots & 1 & \\ & & & 2 & \\ & & & 1 & \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_{2^{m-1}-2} \\ \tilde{x}_{2^{m-1}-1} \end{bmatrix}.$$

Similar to restriction, the one-dimensional prolongation operator from grid size

$\frac{1}{2^m}$ to $\frac{1}{2^{m+1}}$ is defined as

$$I_m^{m+1} = \frac{1}{2}\begin{bmatrix} 1 & & & \\ 2 & & & \\ 1 & 1 & & \\ & 2 & & \\ & 1 & \vdots & \\ & & \vdots & \\ & & \vdots & 1 \\ & & & 2 \\ & & & 1 \end{bmatrix},$$

where its size is $(2^{m+1}-1)$-by-(2^m-1).

4.2.3 The Geometric Multigrid Method

There are two types of multigrid methods, namely geometric and algebraic. The algorithm of geometric multigrid method and its efficiency can be shown by solving the following ODE,

$$\begin{cases} u''(x) = 0, \\ \\ u(0) = u(1) = 0, \end{cases}$$

using central difference approximation as follows.

Suppose that the finest grid size is $\frac{1}{8}$ and coarsest grid size $\frac{1}{2}$. Denote $\mathbf{u}^{(m)}$ be the approximation with grid size $\frac{1}{2^m}$. The discrete problem at the finest grid is

$$64\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}\begin{bmatrix} u_1^{(3)} \\ u_2^{(3)} \\ \vdots \\ u_6^{(3)} \\ u_7^{(3)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}.$$

A pre-smoothing process of k_1 iterations of Jacobi method is performed with the following iterative matrix,

$$\begin{bmatrix} 0 & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{2} & 0 & \frac{1}{2} \\ & & & \frac{1}{2} & 0 \end{bmatrix}.$$

Set $k_1 = 1$ for an illustration, and the initial guess is set to be

$$\left(\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}\right)^T.$$

This initial guess is said to be of low frequency. After one iteration, $\mathbf{u}^{(3)}$ is updated to be

$$\left(\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, \frac{3}{4}, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}\right)^T.$$

Then the residual vector of this solution will be passed to a coarser grid,

$$-\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 & & & & \\ & & 1 & 2 & 1 & & \\ & & & & 1 & 2 & 1 \end{bmatrix} 64 \begin{bmatrix} 2 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & -1 & 2 & -1 & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & -1 \\ & & & & & -1 & 2 \end{bmatrix} \begin{bmatrix} \frac{1}{4} \\ \frac{1}{2} \\ \frac{3}{4} \\ \frac{3}{4} \\ \frac{3}{4} \\ \frac{1}{2} \\ \frac{1}{4} \end{bmatrix} = \begin{bmatrix} -4 \\ -8 \\ -4 \end{bmatrix}.$$

Since now the residual vector $(-4, -8, -4)^T$ is at a grid with grid size $\frac{1}{4}$, the matrix at this grid can be found by using the central difference method with step size $\frac{1}{4}$. This matrix is called the coarse grid matrix. Since this grid is not the coarsest, one iteration of pre-smoothing with initial guess $(0, 0, 0)^T$ is performed

on the following system,

$$16\begin{bmatrix} 2 & -1 & \\ -1 & 2 & -1 \\ & -1 & 2 \end{bmatrix}\begin{bmatrix} u_1^{(2)} \\ u_2^{(2)} \\ u_3^{(2)} \end{bmatrix} = \begin{bmatrix} -4 \\ -8 \\ -4 \end{bmatrix},$$

and $\mathbf{u}^{(2)}$ is updated to be $\left(-\frac{1}{8}, -\frac{1}{4}, -\frac{1}{8}\right)^T$. The residual vector is passed to a coarser grid with grid size $\frac{1}{2}$,

$$\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}\left(\begin{bmatrix} -4 \\ -8 \\ -4 \end{bmatrix} - 16\begin{bmatrix} 2 & -1 & \\ -1 & 2 & -1 \\ & -1 & 2 \end{bmatrix}\begin{bmatrix} -\frac{1}{8} \\ -\frac{1}{4} \\ -\frac{1}{8} \end{bmatrix}\right) = \begin{bmatrix} -4 \end{bmatrix}.$$

Now it is at the coarsest grid, the coarse grid matrix is found by the central difference method with step size $\frac{1}{2}$, and the equation is solved exactly, i.e., we solve

$$8u_1^{(1)} = -4,$$

which gives $u_1^{(1)} = -\frac{1}{2}$. This solution is passed to a finer grid and added to $\mathbf{u}^{(2)}$, i.e., $\mathbf{u}^{(2)}$ is updated to be

$$\begin{bmatrix} -\frac{1}{8} \\ -\frac{1}{4} \\ -\frac{1}{8} \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}\begin{bmatrix} -\frac{1}{2} \end{bmatrix} = \begin{bmatrix} -\frac{3}{8} \\ -\frac{3}{4} \\ -\frac{3}{8} \end{bmatrix}.$$

Use the updated $\mathbf{u}^{(2)}$ as an initial guess to perform one more time of Jacobi method, i.e., $\mathbf{u}^{(2)}$ is updated to be

$$\begin{bmatrix} 0 & \frac{1}{2} & \\ \frac{1}{2} & 0 & \frac{1}{2} \\ & \frac{1}{2} & 0 \end{bmatrix}\begin{bmatrix} -\frac{3}{8} \\ -\frac{3}{4} \\ -\frac{3}{8} \end{bmatrix} + \frac{1}{32}\begin{bmatrix} -4 \\ -8 \\ -4 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \end{bmatrix}.$$

This process is called post-smoothing. Then this updated $\mathbf{u}^{(2)}$ is passed to a finer

grid and added to $\mathbf{u}^{(3)}$, i.e., $\mathbf{u}^{(3)}$ is updated to be

$$\begin{bmatrix} \frac{1}{4} \\ \frac{1}{2} \\ \frac{3}{4} \\ \frac{3}{4} \\ \frac{3}{4} \\ \frac{1}{2} \\ \frac{1}{4} \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 1 & & & & & & \\ 2 & & & & & & \\ 1 & 1 & & & & & \\ & 2 & & & & & \\ & 1 & 1 & & & & \\ & & & 2 & & & \\ & & & & 1 & & \end{bmatrix} \begin{bmatrix} -\frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \\ 0 \\ 0 \end{bmatrix}.$$

Using this as an initial guess to perform one more iteration will give

$$\left(0, \frac{1}{8}, \frac{1}{8}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}, 0\right)^T,$$

which is the output of one complete V-cycle. If one more V-cycle is to be performed, this output should be used as the initial guess of the next V-cycle and so on. Since the zero function is the solution of the ODE, the max norm of the error after one V-cycle is $\frac{1}{4}$. If the discrete problem at the finest grid is solved by the Jacobi method, the approximation after 5 iterations is

$$\left(\frac{7}{32}, \frac{3}{8}, \frac{17}{32}, \frac{17}{32}, \frac{17}{32}, \frac{3}{8}, \frac{7}{32}\right)^T,$$

with the error $\frac{17}{32}$. The problem of Jacobi method is that it can handle high frequency error well but not for low frequency error. As a result, most of the iterations are devoted to handling the low frequency error. Using multigrid method, low frequency error can be passed to a coarser grid and become high frequency error, then the iterative method can work well at the coarser grid. The geometric multigrid method is summarized as follows.

Fix the finest grid size to be $\frac{1}{2^{m_2}}$, and the coarsest grid size $\frac{1}{2^{m_1}}$. Let $A^{(m)}$ be the coarse grid matrix at the grid with grid size $\frac{1}{2^m}$. Suppose also that the problem at the finest grid is

$$A^{(m_2)}\mathbf{x} = \mathbf{b}^{(m_2)}. \tag{4.2}$$

Then the following steps describe the V-cycle algorithm.

1. Apply k_1 Jacobi iterations on (4.2) using last V-cycle output as the initial guess, and choose an arbitrary initial guess for the first V-cycle. The approximation is denoted by $\mathbf{u}^{(m_2)}$.

2. For m from m_2 to $m_1 + 1$, compute the residual vector

$$\mathbf{r}^{(m)} = \mathbf{b}^{(m)} - A^{(m)}\mathbf{u}^{(m)},$$

and pass it to a coarser grid, denoted by $\mathbf{b}^{(m-1)}$,

$$\mathbf{b}^{(m-1)} = I_m^{m-1}\mathbf{r}^{(m)}.$$

If $m \neq m_1 + 1$, apply k_1 Jacobi iterations using arbitrary initial guess for

$$A^{(m-1)}\mathbf{x} = \mathbf{b}^{(m-1)},$$

and the approximation is denoted by $\mathbf{u}^{(m-1)}$, update $m \rightarrow m-1$ and repeat the process.

If $m = m_1 + 1$, find $\mathbf{u}^{(m_1)}$ such that

$$A^{(m_1)}\mathbf{u}^{(m_1)} = \mathbf{b}^{(m_1)}.$$

3. Then for m from m_1 to $m_2 - 1$, update $\mathbf{u}^{(m+1)}$ to be

$$\mathbf{u}^{(m+1)} \rightarrow \mathbf{u}^{(m+1)} + I_m^{m+1}\mathbf{u}^{(m)}.$$

$\mathbf{u}^{(m+1)}$ is further updated by applying k_1 Jacobi iterations on

$$A^{(m+1)}\mathbf{x} = \mathbf{b}^{(m+1)},$$

using $\mathbf{u}^{(m+1)}$ as an initial guess. Then update $m \rightarrow m + 1$ and repeat the process. The output $\mathbf{u}^{(m_2)}$ is the approximation after one V-cycle.

4.2.4 The Algebraic Multigrid Method

The difference between the geometric and algebraic multigrid method is how to define the coarse grid matrix at a coarser grid. The geometric multigrid method reformulates the problem by discretizing the original ODE with a new step size. The algebraic multigrid method, on the other hand, defines the coarse grid matrix $A^{(m)}$ at the grid with grid size $\frac{1}{2^m}$ to be

$$A^{(m)} = I_{m+1}^{m} I_{m+2}^{m+1} \cdots I_{m_2}^{m_2-1} A^{(m_2)} I_{m_2-1}^{m_2} \cdots I_{m+1}^{m+2} I_{m}^{m+1}$$

where $m_1 \leq m < m_2$. The algebraic multigrid method coincides with the geometric multigrid method except for this definition of the coarse grid matrix.

4.2.5 Higher Dimensional Cases

The multigrid methods can be generalized to highly dimensional cases using tensor product. For example, the prolongation operator is defined as

$$I_m^{m+1} \otimes I_m^{m+1},$$

and the restriction operator is defined as

$$I_{m+1}^{m} \otimes I_{m+1}^{m},$$

with regard to the definition in §4.2.2.

The two dimensional coarse grid matrix at the grid with grid size $\frac{1}{2^m}$ is

$$(I_{m+1}^{m} \otimes I_{m+1}^{m}) \cdots (I_{m_2}^{m_2-1} \otimes I_{m_2}^{m_2-1}) A^{(m_2)} (I_{m_2-1}^{m_2} \otimes I_{m_2-1}^{m_2}) \cdots (I_m^{m+1} \otimes I_m^{m+1}).$$

4.2.6 Applications in Queueing Networks

For the 2-queue overflow network in (2.5), (3.6) becomes

$$[A \otimes I_{n_1} \otimes I_{n_2} + hI_N \otimes (G_1 \otimes I_{n_2} + I_{n_1} \otimes G_2 + \mathbf{e}_{n_1} \mathbf{e}_{n_1}^T \otimes R_1)]\mathbf{x} = -\mathbf{a} \otimes \mathbf{p}_0. \quad (4.3)$$

Here we describe our method for solving (4.3). Let $A = PDP^{-1}$ be the spectral decomposition of A with D being a diagonal matrix with diagonal entries ρ_i, $1 \leq i \leq N$, where ρ_i are complex numbers with positive real part, see for instance [2, Figure 5.2].

Let $\mathbf{y} = (P^{-1} \otimes I_{n_1} \otimes I_{n_2})\mathbf{x}$. Then (4.3) becomes

$$[D \otimes I_{n_1} \otimes I_{n_2} + hI_N \otimes (G_1 \otimes I_{n_2} + I_{n_1} \otimes G_2 + \mathbf{e}_{n_1}\mathbf{e}_{n_1}^T \otimes R_1)]\mathbf{y} = -P^{-1}\mathbf{a} \otimes \mathbf{p}_0.$$

We decompose this system of equations into N sub-systems of smaller size:

$$[\rho_i(I_{n_1} \otimes I_{n_2}) + h(G_1 \otimes I_{n_2} + I_{n_1} \otimes G_2 + \mathbf{e}_{n_1}\mathbf{e}_{n_1}^T \otimes R_1)]\mathbf{y}_i = c_i\mathbf{p}_0, \quad 1 \leq i \leq N, \tag{4.4}$$

where c_i is the ith entry of $-P^{-1}\mathbf{a}$ and $\mathbf{y}^T = (\mathbf{y}_1^T, \mathbf{y}_2^T, \ldots, \mathbf{y}_N^T)$.

In [7], an AMG method has been used successfully to solve a system of the form

$$(G_1 \otimes I_{n_2} + I_{n_1} \otimes G_2 + \mathbf{e}_{n_1}\mathbf{e}_{n_1}^T \otimes R_1)\mathbf{x} = \mathbf{b},$$

which is the system corresponds to the steady-state queues, and is equal to the transient system in (4.4) but without the first term. Here we will use the same AMG method to solve (4.4).

Suppose in the finest grid we have $(2^{m_2} - 1)$ equations and at the coarsest grid we have $(2^{m_1} - 1)$ equations. We use the V-cycle algorithm here with one pre-smoothing and one post-smoothing at each grid. The smoother we used is the Gauss-Seidel iterative method [13, p. 49].

The prolongation operator I_m^{m+1} is equal to $2(I_{m+1}^m)^T$. For queueing networks, we use the modified restriction operator $\tilde{I}_{m+1}^m$ described in [7] which is defined as follows. Suppose $I_{m+1}^m = (c_{i,j})$, then $\tilde{I}_{m+1}^m \equiv (\tilde{c}_{i,j})$ where

$$\tilde{c}_{i,j} = \frac{c_{i,j}}{d_j}, \quad \text{and} \quad d_j = \sum_i c_{i,j}.$$

If A^m is the matrix at grid m, i.e., there are $(2^m - 1)$ equations, then at grid $m-1$, the matrix A^{m-1} satisfies $A^{m-1} = \tilde{I}_m^{m-1} A^m I_{m-1}^m$. The reason for using the

modified restriction operator is to keep the singularity as well as the structure of G_1 and G_2, see [7] for more details.

For two-queue systems, the prolongation operator and the modified restriction operator are $I_m^{m+1} \otimes I_m^{m+1}$ and $\tilde{I}_{m+1}^m \otimes \tilde{I}_{m+1}^m$ respectively, and the two dimensional coarse grid matrix can be defined accordingly. Note that the coarse grid matrix of an identity matrix is a matrix with 3 bands. We will use this fact in the cost analysis.

Chapter 5

Numerical Experiments

In this section, we first compare the cost of using IVMs and BVMs for solving overflow queues. Then we illustrate the efficiency of our method on three overflow queues and two networks with batch arrival. A comparison of the AMG method and the GMRES method [17] will also be given.

Consider a general two-queue network with Queue i having n_i states. For a general IVM, in each time step, it requires only a few matrix-vector multiplications which are of order $O(n_1 n_2)$ as the matrix H in (3.4) is a banded matrix of size $n_1 n_2$-by-$n_1 n_2$ with 5 bands. If h_c is the largest time step guaranteed for convergence, then the IVM will require T/h_c's time steps to get the final time T. But in order for the method to converge, h_c should be of order $O(1/\|H\|)$. Thus the total cost of the IVM is of $O(n_1 n_2 \|H\| T)$.

For a general BVM, there are T/h's sub-systems to be solved. We solve each sub-system by the AMG method. The cost for each V-cycle is of $O(n_1 n_2)$ operations since the coefficient matrix of each sub-system is of size $n_1 n_2$-by-$n_1 n_2$ and has at most 9 bands. Thus the total cost is about $O(n_1 n_2 k T/h)$, where k is the maximum of the numbers of V-cycles required for convergence for each

sub-system.

We remark that if $\|H\|$ is bounded independent of n_i, then both methods will be of the same order. However, if $\|H\|$ is increasing with n_i, then the BVM will be less costly than the IVM. We illustrate this by three examples. The first example describes the situation that the arrival and service rate are independent of the size of the queues. The second and the third examples, on the other hand, describe the situations with arrival and service rate dependent on the size of the queues.

Example 1: For $i = 1, 2$, let G_i^M be the same $(2^M - 1)$-by-$(2^M - 1)$ matrix as in (2.4) with $s_i = 5$, $\mu_i = 1$, $\lambda_i = s_i\mu_i - \frac{1}{2}(n_i - 1)^{-1} = 5 - \frac{1}{2}(2^M - 2)^{-1}$.

Example 2: For $i = 1, 2$, let G_i^M be the $(2^M - 1)$-by-$(2^M - 1)$ matrix,

$$G_i^M = \begin{bmatrix} 2^{M-1} & -2^{M-1} & & & 0 \\ -2^{M-1} & 2^M & -2^{M-1} & & \\ & \ddots & \ddots & \ddots & \\ & & -2^{M-1} & 2^M & -2^{M-1} \\ 0 & & & -2^{M-1} & 2^{M-1} \end{bmatrix}.$$

Example 3: For $i = 1, 2$, let G_i^M be the same $(2^M - 1)$-by-$(2^M - 1)$ matrix as in (2.4) with $\mu_i = 2^{M-1}$, $s_i = 5$, $\lambda_i = s_i\mu_i - \frac{1}{2}(n_i - 1)^{-1} = 5 \cdot 2^{M-1} - \frac{1}{2}(2^M - 2)^{-1}$.

In all examples, we solve for the probability distribution vector at $T = 10$. We assume that the initial state is $(0, 0)$. In solving (4.4) by the AMG method or the GMRES method, we use a stopping tolerance of 10^{-6}. For the AMG method, we set m_1, the coarsest grid level, to be 2.

In Example 1, $\|G_i^M\|$ and hence $\|H\|$ is bounded. Thus, the total costs for both the IVM and the BVM are of order $O(2^{2M})$. In Examples 2 and 3, $\|H\|$ is of order $O(2^M)$. So for the IVM, the total cost is $O(2^{3M})$. But for the BVM, h can be kept constant regardless of $\|H\|$. Thus the total cost for each V-cycle is still of order $O(2^{2M})$. To be more specific, in the following we estimate the total costs for RK4 and GBDF3 in terms of number of scalar multiplications.

For RK4, 4 matrix-vector multiplications, 4 vector-vector operations, and 6 scalar-vector multiplications are required in each time step. The matrix involved is a 5-band matrix of size $(2^M-1)^2$-by-$(2^M-1)^2$. Thus

$$4 \cdot 10(2^M-1)^2 + 4(2^M-1)^2 + 6(2^M-1)^2 = 50(2^M-1)^2$$

operations are required in each time step. In Example 1, $h_c = 0.25$. Since $T/h_c = 40$ time steps are required, the total cost is $(2000 \cdot (2^M-1)^2)$ operations. In Example 2, since $(10 \cdot 2^{M+1})$ time steps are required, the total cost is $(1000 \cdot 2^M(2^M-1)^2)$ operations. In Example 3, since $(10 \cdot 2^{M+3})$ time steps are required, the total cost is $(4000 \cdot 2^M(2^M-1)^2)$.

For GBDF3 with the AMG method, we first estimate the cost for each V-cycle. As the cost of a Gauss-Seidel iteration for a 9-band matrix is 18 operations for each update of a variable, the cost for each V-cycle is

$$2\sum_{i=2}^{M} 18(2^i-1)^2 \leq 36\sum_{i=2}^{M}(4^i) \leq 12 \cdot 4^{M+1} = 48 \cdot 2^{2M}$$

operations. We can take $h = 1$ for all three examples as the method is stable. Thus there are 10 sub-systems to be solved by the AMG solvers. The total cost is therefore less than $(480k \cdot 2^{2M})$ operations.

Tables 5.1 to 5.3 give h_c, k, the total number of operations as estimated above, and the CPU times in solving the problems by Matlab. We can see that the BVM together with the AMG method is more efficient in finding the transient solutions when compared with the IVM. We see also that the AMG method is very efficient in solving these systems—the number of V-cycles required for convergence is independent of the queue sizes even when $\|H\|$ increases.

Regarding Examples 2 and 3, it seems that the matrix G_i^M in these two examples can be made better conditioned by scaling down by the factor 2^{M-1} as follows: divide 2^{M-1} from both sides in the ODE (3.1) and let $\tilde{t} = 2^{M-1}t$, then

we have

$$\frac{d\mathbf{p}(\tilde{t})}{d\tilde{t}} = -\tilde{H}\mathbf{p}(\tilde{t}),$$

where $\|\tilde{H}\|$ is independent of M. However, in order to find $\mathbf{p}(t)$ at time T, one has to find $\mathbf{p}(\tilde{t})$ at time $\tilde{T} = 2^{M-1}T$. By the previous results, the total cost for the IVMs on this new ODE is of order $O(n_1 n_2 \|\tilde{H}\|\tilde{T}) = O(n_1 n_2 \|H\|T)$. That means the order of the total cost cannot be reduced by scaling.

After that, we compare the AMG method with the preconditioned GMRES method [17]. The preconditioner for (4.4) is

$$\rho_i(I_{n_1} \otimes I_{n_2}) + h(c(G_1) \otimes I_{n_2} + I_{n_1} \otimes c(G_2)).$$

Table 5.4 gives the maximum number of iterations for the (preconditioned) GMRES method over each sub-system in (4.4). We note that the cost per iteration of the GMRES method is of the same order as the AMG method, as it requires mainly matrix-vector multiplications. However, as we can see from Table 5.4, its number of iterations required for convergence increases with the size of queues for Examples 2 and 3. Hence it is more costly than the AMG method.

The BVM with the AMG method also works for the networks with batch arrival. Consider a one-dimensional batch arrival problem as follows.

Example 4: Take

$$\tilde{\lambda}_j = \frac{1}{2^j},$$

in (2.6). So

$$\tilde{\lambda} = \sum_{k=1}^{\infty} \tilde{\lambda}_k = 1.$$

Example 5: Take

$$\tilde{\lambda}_j = \frac{\sqrt{2^m - 1}}{2^j},$$

in (2.6), thus

$$\tilde{\lambda} = \sum_{k=1}^{\infty} \tilde{\lambda}_k = \sqrt{2^m - 1}.$$

In these two examples, $\mu = \dfrac{\tilde{\lambda}}{s}$.

The Kolmogorov backward equation for the batch arrival is again discretized by using the GBDF3, and the resulting linear system is solved by the AMG method. The ODE is also solved by RK4 as a comparison. Table 5.5 shows h_c for RK4 and k for the GBDF3 for Example 4 and Example 5 for $s = 1$, and Table 5.6 gives the results for $s = 5$. The numerical results show that k is independent of m but h_c decreases as m increases for Example 5.

Table 5.1: Total number of operations and CPU times in seconds for Example 1.

	IVM			BVM		
M	h_c	Operations	CPU time	k	Operations	CPU time
3	2^{-2}	98,000	0.49	7	215,040	1.00
4	2^{-2}	450,000	2.37	7	860,160	5.85
5	2^{-2}	1,922,000	10.81	7	3,440,640	28.26
6	2^{-2}	7,938,000	49.05	7	13,762,560	111.71
7	2^{-2}	32,258,000	222.06	7	55,050,240	546.06

Table 5.2: Total number of operations and CPU times in seconds for Example 2.

	IVM			BVM	
M	h_c	Operations	CPU time	k	Operations	CPU time
3	2^{-4}	392,000	1.91	8	245,760	1.13
4	2^{-5}	3,600,000	18.66	8	983,040	6.49
5	2^{-6}	30,750,000	170.70	8	3,932,160	32.24
6	2^{-7}	250,016,000	1,649.07	9	17,694,720	161.99
7	2^{-8}	2,064,512,000	19,131.92	9	70,778,880	792.72

Table 5.3: Total number of operations and CPU times in seconds for Example 3.

	IVM			BVM	
M	h_c	Operations	CPU time	k	Operations	CPU time
3	2^{-6}	1,568,000	7.62	8	245,760	1.13
4	2^{-7}	14,400,000	75.02	8	983,040	6.66
5	2^{-8}	123,000,000	693.18	8	3,932,160	32.26
6	2^{-9}	1,000,064,000	6,634.67	9	17,694,720	166.58
7	2^{-10}	8,258,048,000	>50,000	9	70,778,880	819.20

Table 5.4: Maximum number of iterations needed for the GMRES method.

	No preconditioner			T. Chan's preconditioner		
M	Ex. 1	Ex. 2	Ex. 3	Ex. 1	Ex. 2	Ex. 3
3	27	30	30	19	20	21
4	50	58	69	26	32	35
5	56	89	133	31	51	59
6	54	132	244	28	79	102
7	54	168	>300	26	123	179

Table 5.5: Results for Example 4 and Example 5 for $s = 1$.

	Example 4.		Example 5.	
	IVM	BVM	IVM	BVM
M	h_c	k	h_c	k
3	2^{-1}	4	2^{-2}	8
4	2^{-1}	5	2^{-3}	8
5	2^{-1}	4	2^{-3}	8
6	2^{-1}	4	2^{-4}	7
7	2^{-1}	4	2^{-4}	7

Table 5.6: Results for Example 4 and Example 5 for $s = 5$.

	Example 4.		Example 5.	
	IVM	BVM	IVM	BVM
M	h_c	k	h_c	k
3	2^{-1}	4	2^{-2}	6
4	2^{-1}	5	2^{-3}	8
5	2^{-1}	4	2^{-3}	8
6	2^{-1}	4	2^{-4}	7
7	2^{-1}	4	2^{-4}	6

Chapter 6

Concluding Remarks

In this thesis, the AMG method is used to solve the resulting linear system from the BVMs. A comparison for the IVMs and the BVMs with the AMG method is given. Some networks like two-queue overflow networks and networks with batch arrival have been tested. The numerical results illustrate that the BVMs with the AMG method can be satisfactorily applied to queueing networks, even the problem is ill-conditioned. Further research on applications to other queueing networks can be conducted. In addition, the numerical results also support the convergence of the methods, of which the convergence in finding the steady-state solutions is already proved in [7].

Bibliography

[1] D. Braess, *Towards Algebraic Multigrid for Elliptic Problems of Second Order*, Computing, 55 (1995), 379–393.

[2] L. Brugnano, and D. Trigiante, *Solving Differential Problems by Multistep Initial and Boundary Value Methods*, Gordon and Breach Science Publishers, 1998.

[3] R. Chan, *Iterative Methods for Overflow Queueing Models I*, Numer. Math., 51 (1987), 143–180.

[4] R. Chan, *Iterative Methods for Overflow Queueing Models II*, Numer. Math., 54 (1988), 57–78.

[5] R. Chan and W. Ching, *A Direct Method for Stochastic Automata Networks*, Proceedings of the IMS Workshop on Applied Probability, 1–18, Hong Kong, Eds.: R. Chan, Y. Kwok, D. Yao, and Q. Zhang, Studies in Advanced Mathematics, 26, International Press, June 1999.

[6] R. Chan and W. Ching, *Circulant Preconditioners for Stochastic Automata Networks*, Numer. Math., 87 (2000), 35–57.

[7] Q. Chang, S. Ma, and G. Lei, *Algebraic Multigrid Method for Queuing Networks*, Inter. J. Comput. Math., 70 (1999), 539–552.

[8] W. Ching, *Iterative Methods for Queuing and Manufacturing Systems*, Springer Monographs in Mathematics, Springer, London, 2001.

[9] W. Ching, R. Chan, and X. Zhou, *Circulant Preconditioners for Markov Modulated Poisson Processes and Their Applications to Manufacturing Systems*, SIAM J. Matrix Anal. Appl., 18 (1997), 464–481.

[10] W. Ching and A. Loh, *Iterative Methods for Flexible Manufacturing Systems*, Appl. Math. Comput., 141 (2003), 553–564.

[11] W. Ching, W. Yuen, and A. Loh, *An Inventory Model with Returns and Lateral Transshipments*, J. Operat. Res. Soc., 54 (2003), 636–641.

[12] J. Climent, L. Tortosa and A. Zamora, *A Note on the Recursive Decoupling Method for Solving Tridiagonal Linear Systems*, Appl. Math. Comput., 140 (2003), 159–164.

[13] W. Hackbusch, *Multi-Grid Methods and Applications*, Springer-Verlag Berlin, 1985.

[14] L. Lu, W. Ching, and M. Ng, *Exact Algorithms for Singular Tridiagonal Systems with Applications to Markov Chains*, to appear in Appl. Math. Comput. (2004).

[15] L. Kaufman, *Matrix Methods for Queueing Problems*, SIAM J. Statist. Comput., 4 (1982), 525–552.

[16] R. Kimmel and I. Yavneh, *An Algebraic Multigrid Approach for Image Analysis*, SIAM J. Sci. Comput., 24 (2003), 1218–1231.

[17] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, 2003.

[18] R. Sideje and W. Stewart, *A Numerical Study of Large Sparse Matrix Exponetials Arising in Markov Chain*, Comput. Stat. Data Anal., 29 (1999), 345–368.

[19] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag New York Inc., 1980.

[20] K. Stuben, *Algebraic Multigrid (AMG): Experiences and Comparisons*, Appl. Math. Comput., 13 (1983), 3–4, 419–451.

[21] H. Taha, *Operations Research: An Introduction*, Prentice Hall, 1997.