An Iterative Algorithm for Inversion of Matrices

Jayashree Rajagopalan

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfilment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montreal, Quebec, Canada

September 1996

ISBN 0-612-18429-3

Canada

ABSTRACT

An Iterative Algorithm for Inversion of Matrices

Jayashree Rajagopalan

Matrix computations are the backbone of many science and engineering problems. Matrix inversion is used in many real-time engineering applications which have stringent computational requirements. It has been mentioned in the literature that iterative procedures are suitable to manipulate large matrices as they are robust and less susceptible to numerical errors compared to direct methods.

This thesis focuses on the development of an improved iterative matrix inversion scheme. A generalized, efficient matrix inversion scheme has been developed. The proposed scheme requires $n^2(k-1)$ less operations than the conventional approach, where n is the order of the matrix and k is the total number of iterations. The computing efficiency of the proposed scheme is further improved by optimizing the number of terms from the infinite series used in the computation. Investigation of the analytical results reveals that the optimum choice for the number of terms from the power series is 3 for a fixed number of arithmetic operations. Iterative schemes require an approximate initial solution which is successively improved till the desired solution is obtained, and a stopping criterion to limit the number of iterations to be carried out. Three schemes to compute the approximate initial solution using the one, infinity and Frobenius norms have been developed. A stopping criterion has been proposed to determine the number of iterations needed to compute the inverse.

It has been experimentally substantiated that fastest convergence to the desired inverse for a fixed amount of operations indeed requires only three terms from the infinite series. Tests have been carried out for the proposed iterative scheme using the developed initial inverse formulations to validate the analytical formulations. Experiments carried out with matrices of different orders indicate that the convergence is not very sensitive to the orders of the matrices.

ACKNOWLEDGEMENT

I would like to express my deep sense of gratitude to my supervisor, Prof. M.O. Ahmad for his guidance, patience and understanding. Without his goodwill this thesis would not have been possible. I am indebted to him for supervising my thesis work.

My sincere thanks are also due to Prof. R.M.H. Cheng for having been very helpful on numerous occasions.

I have had immeasurable support from my husband, Ramesh, without whose encouragement, understanding and constant help I would not have completed this work. He has been the pillar of strength behind my efforts. My thanks are also due to my son, Aditya, for his wonderful sense of understanding at such an early age, which has been of great help in my efforts. My thanks are also due to my entire family for having made all of this possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

λ_s	Spectral Radius
$\phi(\lambda)$	Characteristic polynomial
$\| A \|_2$	Spectral norm of matrix A
$\| A \|_1$	One norm of matrix A
A^{-1}	Inverse of matrix A
A^T	Transpose of matrix A
cond(A), K(A)	Condition number of matrix A with respect to some norm
det(A)	Determinant of a matrix A
E_0	Initial residual matrix
$K_1(A)$	Condition number of matrix A with respect to one norm
$K_F(A)$	Condition number of matrix A with respect to Frobenius norm
N_k	Number of operations in the kth iteration
N	Number of operations for k iterations
p	Number of terms chosen in the power series
q	Norm of initial residual matrix E_0
R_k	Computed inverse in the kth iteration
$\lceil x \rceil$	Ceiling of x : denotes the least integer equal to or greater than x

CHAPTER 1

INTRODUCTION

1.1 General

A huge amount of computer resources is spent for carrying out matrix computations for solving linear systems, which is the backbone of many science and engineering problems. The connection between digital computation and matrices is obvious. Matrices represent linear transformations from one finite set of numbers to another finite set. Since many engineering problems are linear, and since digital computers with finite memory can manipulate only finite sets of numbers, the solution of linear problems by digital computation usually involves matrices.

Matrix theory is used in the state variable representation of a system [1]. The state variable representation of a system is concerned with a system as a whole, including the internal variables of the system, and the input/output variables. These are characterized by a state matrix. In solving for these variables, matrix inversion is inevitable. The characterization by state matrices is important in studying complex systems, their stability, determining critical parameters for achieving improved performance and also in many signal processing applications. For example, to find the impulse response of a discrete time system, the state variable representation of the system can be used. The state variable representation of the system in turn is given in matrix

notation. Calculations involving these matrices invariably involve the inversion of matrices. Real-time signal processing applications have a stringent requirement of carrying out computations within a minimum amount of time. In some cases, information continues to arrive and must be incorporated into the solution or updated while in some others it may be necessary to remove old observations from the solutions. Applications involving recursive least square updating or downdating include robust regression in statistics, optimization techniques, and estimation techniques in adaptive signal processing and control [2]. There are many scientific and industrial applications which require enormous processing power. In some cases huge amounts of data must be processed to produce usable results, and in others, large amounts of calculations have to be carried out on small amount of data.

1.2 Applications of Matrix Inversion

In general solution of linear systems require matrix inversion operations. For example, image reconstruction problems require operations of matrix inversion, which may be time consuming if a very accurate estimate is needed. Image reconstruction using singular value decomposition and a neural network method are discussed by Steriti and Fiddy [3] for finding the inverse of a matrix for the prioritized discrete fourier transform (PDFT). The PDFT algorithm is used to recover an estimate of the original image f from the available data g given by [3]

$$g(x) = \int A(x,y) f(y)\, dy + k \tag{1.1}$$

where A is the transformation kernel and k is a noise signal. If the data is spectral data, this can be given as [3]

$$g = Af + k \tag{1.2}$$

where g and f are vectors and A is the transformation matrix. To estimate f, the inverse of A has to be calculated.

Another application requiring inversion of matrices using iterative scheme is phased-array radar [4]. Target tracking is a recursive prediction correction process, where Kalman filtering [5] is extensively used. Target equations are modelled explicitly such that the position, velocity and potentially higher derivatives of each measurement are estimated by the track filter as a state vector. The error estimated with the state vector is modelled using a covariance matrix, which is then used in subsequent computations, where matrix inversion is one such computation. This covariance matrix gets updated in each iteration of the track filter. Finding the inverse in the next iteration could make use of the inverse in the present iteration. In such cases, the matrices involved are very large, and efficient iterative inverse procedures are important to ensure fast computations.

In VLSI circuit simulation, large systems of equations have to be solved. To solve these, the system is partitioned into groups of tightly coupled variables, called blocks, such as the subcircuits in the VLSI structure. These matrices represent the internal variables of a subcircuit, and the interconnections among the subcircuits. A method to

invert such matrices in which the matrices are in a block diagonal form is discussed by Milovanovic et al. [6].

Another real-time application is that of noise cancellation from the main signal as discussed by Welstead [7]. The receiver configuration may consist of a main antenna and several omni-directional side antennas. The main antenna receives both the desired signal and the unwanted noise. The side antenna try to pick up the noise alone, so that the noise can be subtracted from the combination of signal and noise to get the desired output. However, the noise received by the side antenna may be a delayed version of what is received by the main antenna. The characterization of delayed versions of the secondary noise signals and its removal from the desired signal invoives the minimization of a function where inversion of the noise covariance matrix must be computed.

Spreadsheet programs make it easy to carry out many calculations on chemical systems and these have been discussed by Wink [8]. Spreadsheet can quickly carry out certain matrix manipulations. The reacting chemical systems can be described by a matrix formulation, and the corresponding chemical equations can be obtained through the application of linear algebra. The essence of this method is the solution of a linear system of the form $\mathbf{Ax} = \mathbf{b}$, where A is the system matrix with N columns for the N chemical substances and N rows for the M elements; n is the stoichiometric matrix, whose columns contain the coefficients for the substances in a balanced chemical equation. The algebraic challenge consists of finding the stoichiometric matrix from the matrix A.

Spreadsheet programs obtain this by the inversion of the matrix A.

Matrix inversion schemes find use in robotic applications. In robotic applications, it is often required to compute the velocities of the robot joints (joint space velocities) for given velocities of the end-effector of the robot (cartesian linear and angular velocities) by inverting the Jacobian matrix [9]. Though the size of these matrices are not large (6x6 for a robot with six degrees of freedom), the speed at which these computations have to be carried out becomes important for real-time control. Though the direct methods can provide the required solution, the stability of the iterative schemes makes them suitable for these applications. Further, if the sampling time is small (say 10 ms), and if the robot is moving at a slow speed, the elements of the Jacobian matrix changes slowly. Iterative matrix inversion methods are suitable for such slow changes in the elements of the input matrix as solutions obtained during the previous sampling instant can be used as the initial approximate inverse to compute the inverse Jacobian for the current sampling instant.

There are many techniques for matrix inversion that have been discussed in the literature. While some of them are more suitable for direct matrix inversion [8, 10-13] as in spreadsheet computations, or computing the inverse of a Jacobian matrix in robotic applications [9], there are some applications, such as in signal processing applications where it is preferable to use iterative methods [7, 14]. It has been reported in the literature [5] that many direct methods of matrix inversion, require high accuracy in the

calculations to obtain proper solutions as they are not tolerant to errors in the computed matrices. In contrast, iterative methods compensates for individual and accumulation of round-off errors as it is a process of successive refinement.

1.3 Literature Review

Finite precision arithmetic introduces round-off errors. These errors become significant if the given problem is computationally very intensive. Hence, in large engineering problems, this factor has to be taken into account when solving computationally intensive problems. Matrix inversion is a very good example of such problems, where round off and truncation errors result in large errors in the inverted matrix. Matrix inversion can be carried out using a number of methods which can be broadly classified as direct and iterative methods.

Direct methods arrive at an exact solution after a finite number of steps. However, this is only true if we have infinite precision. Since only finite precision is used in computations, the solutions are still approximate. They are susceptible to round off errors. The effect of such inaccuracies become magnified for very large problems such as those involved in matrix manipulations.

Iterative methods, on the other hand, improve successive approximations until the solutions converge to the desired result. Iterative methods for matrix computations are widely used, especially for large systems, because these methods tend to be simpler, more

robust to numerical errors, and require less storage compared to the direct methods [16]. The speed of convergence of iterative methods, however, depends on the initial approximation. Hence, the choice of the initial solution is very important to make an iterative method efficient. Improper choice of the initial solution will result in the divergence of the iterative method from the desired solution. Solutions provided by iterative methods are also no doubt approximate solutions. But the effect of truncation and round-off errors is minimized.

The Gauss-Jordan method [6, 15, 17] is a direct method of matrix inversion. The inversion of a matrix by this method involves applying a series of transformations on the rows and columns of the matrix to convert it into an identity matrix. The corresponding operations on a unit matrix, transforms the identity matrix into the inverse of the original matrix. Several variations of this method are discussed in the literature suitable for implementation on a processor array [15, 17]. The principal advantage of the Gauss-Jordan method is that it is very stable. However, it requires two matrices to be stored and manipulated at the same time. This becomes a major disadvantage when very large matrices need to be inverted.

Milovanovic et al. [11] have discussed a variation of the Gauss-Jordan method suitable for implementation on a processor array, which operates in a pipeline fashion. It uses a global scheduling strategy to calculate intermediate results in processors and redistribute them to all the processors. The processors then calculate the columns based

on these results. A direct parallel matrix inversion algorithm based on the Givens plane rotations [11] is discussed by Amwy and Dharmarajan [18]. A parallel Gauss-Jordan algorithm suitable for implementation on pyramidal multiprocessor system has been analyzed by Geus et al. [19].

Schlereth and Schlereth [12] have discussed the use of a transputer-based large multiple node attached parallel processor for solving ultra-large computing problems. The procedure proposed has been designed to solve large sparse, dense or bounded matrices. This parallel processing environment uses very little communication among the processors during the execution of the algorithm. It has been mentioned that there is a linear increase in the processing power as the number of nodes is increased. The matrix inversion scheme uses the Householder's algorithm [20] because of it's inherent stability. Schlereth et al. [12] elaborate the parallel implementation of the Householder's algorithm to reduce the matrix to a triangular form. This results in an approximate solution. The paper also mentions that this approximate result can be improved by an iterative algorithm.

Cholesky method [17], also known as the square-root method can be used to calculate the inverse of a matrix. Consider a system of the form $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a positive definite, symmetric matrix. In the Cholesky method, the given matrix, $\mathbf{A}$, has to be decomposable as

$$A = LL^T \tag{1.3}$$

where $\mathbf{L} = [l_{ij}]$, i,j = 1,...,n and $l_{ij} = 0$, i < j.

The inverse of **A** is given by

$$A = (L^{-1})^T L^{-1} \tag{1.4}$$

Thus, the complexity of finding the inverse of a symmetric matrix has been reduced to the one of finding the inverse of a lower triangular matrix. Since the inverse of a lower triangular matrix is also lower triangular, the complexity of finding the inverse is reduced.

Gianey [21] outlines the use of Strassen's algorithm [22] to compute the matrix inverse on a transputer array. The Strassen's algorithm is a matrix multiplication algorithm in which the two n x n matrices to be multiplied are partitioned into sub-matrices. The product matrix **D** = **EF** is given by

$$\begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix} = \begin{bmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{bmatrix} \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \tag{1.5}$$

where

$\mathbf{D_{11}}, \mathbf{E_{11}}, \mathbf{F_{11}}$, are of the order p x p

$\mathbf{D_{12}}, \mathbf{E_{12}}, \mathbf{F_{12}}$, are of the order p x q

$\mathbf{D_{21}}, \mathbf{E_{21}}, \mathbf{F_{21}}$, are of the order q x p

$\mathbf{D_{22}}, \mathbf{E_{22}}, \mathbf{F_{22}}$, are of the order q x q

$p = \left\lceil \frac{n}{2} \right\rceil$ and q = n - p, where $\lceil\ \rceil$ denotes the least integer equal to or greater than n/2.

The multiplication operation is confined to computing the intermediate quantities X_i, i = 1,.. ,7, where X_i's are the intermediate results obtained by adding and multiplying the sub-matrices of **E** and **F** as given below.

$$\begin{aligned} X_1 &= (E_{11} + E_{22})(F_{11} + F_2) \\ X_2 &= (E_{21} + E_{22})F_{11} \\ X_3 &= E_{11}(F_{12} - F_{22}) \\ X_4 &= E_{22}(F_{21} - F_{11}) \\ X_5 &= (E_{11} + E_{12})F_{22} \\ X_6 &= (E_{21} - E_{11})(F_{11} + F_{12}) \\ X_7 &= (E_{12} - E_{22})(F_{21} + F_{22}) \end{aligned} \tag{1.6}$$

Using the intermediate results X_1 to X_7, the product matrix **D** is computed as shown below:

$$\begin{aligned} D_{11} &= X_1 + X_4 - X_5 + X_7 \\ D_{12} &= X_3 + X_5 \\ D_{13} &= X_2 + X_4 \\ D_{14} &= X_1 + X_3 - X_2 + X_6 \end{aligned} \tag{1.7}$$

Gianey [21] has outlined the use of the Strassen's method [22] to obtain the inverse of a matrix **A**. Let the matrices $B = A^{-1}$ be divided into sub-matrices as shown in (1.8).

$$A^{-1} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^{-1} = B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \tag{1.8}$$

Then intermediate quantities S_1,...,S_6 are computed from the given matrix A, using the equations shown below:

$$\begin{aligned} S_1 &= A_{11}^{-1} \\ S_2 &= A_{21}\ S_1 \\ S_3 &= S_1\ A_{12} \\ S_4 &= A_{21}\ S_3 \\ S_5 &= S_4 - A_{22} \\ S_6 &= S_5^{-1} \end{aligned} \tag{1.9}$$

Using the sub-matrices S_1,...S_6 the sub-matrices B_{11}, B_{12}, B_{21}, B_{22}, of the matrix **B** which is the inverse of **A** can be computed using the relationship given below.

$$\begin{aligned} B_{12} &= S_3\ S_6 \\ B_{21} &= S_6\ S_2 \\ B_{11} &= S_1 - (S_3\ B_{21}) \\ B_{22} &= -S_6 \end{aligned} \tag{1.10}$$

In this method, the matrix multiplications and the inverse is computed recursively and it is suitable for implementation for very large matrices on a parallel system with shared memory. This method is however not suitable for any class of invertible matrices. It can be used for triangular matrices that are invertible and for matrices tnat can be decomposed into an upper and a lower triangular matrix. Further, this method cannot be implemented efficiently on a parallel system with distributed memory.

Pease has outlined a matrix inversion method based on partitioning [23]. The method is based on a two-fold partition of the matrix to be inverted:

where **A**, **B**, **C** and **D** are the sub-matrices into which **M** is partitioned. The inverse of

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{1.11}$$

M can then be obtained as

$$M^{-1} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1}(I-BD^{-1}CA^{-1})^{-1} & -A^{-1}BD^{-1}(I-CA^{-1}BD^{-1})^{-1} \\ -D^{-1}CA^{-1}(I-BD^{-1}CA^{-1})^{-1} & D^{-1}(I-CA^{-1}BD^{-1})^{-1} \end{bmatrix} \tag{1.12}$$

The sub-matrices **A** and **D** are square matrices, but may not necessarily have the same dimensions. If **A** and **D** are not of the same dimension, then **B** and **C** are rectangular. This formula is not applicable if either matrix **A** or **D** is singular. This method can be applied recursively to obtain the inverse. In order to apply this method, we need *a priori* information on the applicability of the method.

In another method of matrix inversion, called the double-bordering method, Levin and Evan [24] have used a process of successive bordering to obtain the inverse of a matrix. This method is suitable for parallel implementation on MIMD computers. However, it is not suitable for all classes of matrices. The matrices must have non-zero principal minors.

A parallel algorithm developed by Csanky for fast matrix inversion has been discussed in [25]. It is based on the Leverrier method [26] for computing the characteristic polynomial of the given matrix **A**. The characteristic polynomial $\phi(\lambda)$ of **A** is defined by

$$\phi(\lambda) = \det(A - \lambda I)$$
$$= (-1)^N (\lambda^N + p_1\lambda^{N-1} + p_2\lambda^{N-2} + ... + p_{N-1}\lambda + p_N) \quad (1.13)$$

The roots $\lambda_1, \lambda_2, ...,\lambda_N$ are the eigenvalues of the matrix **A**. The coefficients p_i, i=1, 2,...,N, of the characteristic equation is the solution of the linear triangular system,

$$T\,p = -t \quad (1.14)$$

where $\mathbf{p} = (p_1, p_2, ... p_n)^t$ and $\mathbf{t} = (t_1, t_2,.....,t_N)^t$, $\mathbf{t_k} = tr(\mathbf{A^k})$, $\mathbf{p} = \mathbf{T^{-1}t}$ and the matrix **T** is be given by

$$T = \begin{bmatrix} 1 & 0 & 0 & .... & 0 & 0 \\ t_1 & 2 & 0 & .... & 0 & 0 \\ t_2 & t_1 & 3 & .... & 0 & 0 \\ . & . & . & .... & . & . \\ . & . & . & .... & . & . \\ . & . & . & .... & . & . \\ t_{N-1} & t_{N-2} & t_{N-3} & .... & t_1 & N \end{bmatrix} \quad (1.15)$$

To invert the given matrix **A**, the values $\mathbf{t_k}$ are first computed. Note that this involves the computation of the powers of **A** and their traces. Then, by using the systems of equations (1.4), the coefficients of the characteristic polynomial are obtained as $\mathbf{p} = -\mathbf{T^{-1}t}$. With the knowledge of the powers of **A** and the vector **p**, the inverse of a matrix **A** can be computed as

$$A^{-1} = \frac{A^{N-1} - p_1 A^{N-2} - p_2 A^{N-3} - \ldots - p_{N-1} I}{p_N} \qquad (1.17)$$

Although Csanky's algorithm is a fast algorithm for matrix inversion, it does not lend itself to practical implementation as it is very unstable [27].

It has been mentioned in the literature [26] that iterative procedures are suitable to manipulate large matrices as they are robust and less susceptible to numerical errors compared to direct methods. The iterative method for matrix inversion developed by Pan and Reif [14, 29] present a formulation that uses only the first two terms of an infinite series involved in representing the inverse. Further, the formulations of initial inverse using various norms and their effect on the convergence of iterative schemes to determine the inverse has not been studied. The effect of the choice of the number of terms, on the error between the computed inverse and the actual inverse, for a fixed amount of computations, has not been examined in the literature.

1.4 Scope and Organization of the Thesis

The focus of this thesis is to develop an efficient iterative matrix inversion algorithm involving an infinite series expansion, to investigate (i) the effect of the number of terms from the infinite series on the convergence to the desired inverse; (ii) the possibility of reducing the number of operations and present a simple generalized iterative formulation to calculate the inverse, and (iii) the development of a simple scheme to choose the initial inverse for the iterative procedure.

From the computational point of view, it is essential to reduce the number of operations in order to devise efficient algorithms. Chapter 2 proposes a new simple formulation to calculate the inverse iteratively involving a reduced number of operations. The efficiency of the algorithm is further improved in the case of a fixed number of arithmetic operations by choosing an optimum number of terms from the power series used in the inverse calculations. A proof for the choice of an optimum number of terms from the power series expansion is derived. Iterative schemes for matrix inversion require the computation of the initial inverse by some means. Improper choice of the initial inverse will result in failure to arrive at the desired inverse. Formulations of the initial inverses using matrix norms are also derived in this chapter Since iterative schemes need a stopping criterion to determine the number of iterations that have to be carried out, a stopping criterion is also proposed.

Chapter 3 elaborates the results of the tests carried out to experimentally to verify the theory developed in Chapter 2. A T-800 transputer along with Occam, a parallel language, is used for this purpose. Results of the tests are presented to verify that the new simplified iterative formulation developed in Chapter 2 converges to the desired inverse. Experimental verification for the optimal choice of the terms from the infinite series is carried out. Test results to verify the correctness of the initial inverse formulations are included. The effect of the choice of different initial inverses formulated in this thesis is also investigated. Conclusion of the present study and the scope for future investigation are presented in Chapter 4.

CHAPTER 2

ANALYTICAL FORMULATIONS FOR ITERATIVE MATRIX INVERSION

2.1 Introduction

Iterative matrix inversion schemes involve successive improvements of the initial inverse calculated. These methods also need a stopping criterion to determine whether the desired inverse has been obtained. Thus, iterative schemes for matrix inversion have three components, namely, (1) a scheme to compute the initial inverse, (2) a scheme to successively improve the inverse starting from the initial inverse, and (3) a stopping criterion to determine if the desired solution has been achieved. In this chapter, a formulation for matrix inversion that yields a good approximation of the desired inverse for a fixed amount of computations is proposed. This approach is based on the Pan and Reif method [14, 29]. A proof is provided to show the validity of the formulation. A method to determine the initial inverse for a general dense matrix, and a stopping criterion for the algorithm are also proposed.

2.2 Matrix Norms and Some Related Definitions

Some of the matrix norms definitions commonly used in matrix algebra are discussed in this section.

Definition: A function $\| . \| : \mathbf{M}_n \rightarrow \mathbf{R}$ is a matrix norm if for all $\{\mathbf{A}, \mathbf{B}\} \in \mathbf{M}_n$ it satisfies

the following five axioms [33]:

1. $\| \mathbf{A} \| >= 0$ (Non-negativity)

2. $\| \mathbf{A} \| = 0$ iff $\mathbf{A} = \mathbf{0}$ (Positivity)

3. $\| cA \| = |c| \| \mathbf{A} \|$ for all complex scalars c (Homogeneous)

4. $\| \mathbf{A} + \mathbf{B} \| <= \| \mathbf{A} \| + \| \mathbf{B} \|$ (Triangle inequality)

5. $\| \mathbf{AB} \| <= \| \mathbf{A} \| \| \mathbf{B} \|$ (Sub-multiplicative)

Some commonly used norms are the one norm, infinity norm, spectral norm and the Frobenius norm. They are defined as follows [28].

One Norm: The maximum column or the one norm as it is called is defined by

$$\|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij}| \tag{2.1}$$

Infinity Norm: The maximum row sum norm as it is otherwise called is defined as

$$\|A\|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}| \tag{2.2}$$

Frobenius Norm: The Frobenius norm or the Schur norm or the Hilbert-Schmidt norm is given by

$$\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2} \tag{2.3}$$

Spectral Norm: The spectral norm is defined as

$$\| A \|_2 = \max \{ \sqrt{\lambda} : \lambda \text{ is an eigenvalue of } A^* A \} \tag{2.4}$$

Spectral Radius: The spectral radius $\rho(A)$ of a matrix A is given by

$$\rho(A) = \max\{ | \lambda | : \lambda \text{ is an eigenvalue of } A \} \tag{2.5}$$

Convergent Matrices: Any square matrix **A** satisfying the condition given below is said to be a convergent matrix [15]

$$\lim_{\gamma \to \infty} A^\gamma = 0 \tag{2.6}$$

Given a matrix A, the following statements are equivalent:

(a) A is convergent;

(b) $\lim_{\gamma \to \infty} \| A^\gamma \| = 0$, for some matrix norm;

(c) $\rho(A) < 1$, where $\rho(A)$ is the spectral radius of the matrix A. The spectral radius is the largest eigen-value of the matrix A.

Relationship Between Commonly Used Norms: In general, $\|A\|_\alpha \leq C_M \|A\|_\beta$. For the values of C_M given in Table 2.1, this inequality is always ensured. It can also be said that the values of C_M given in this table are the maximum values of the ratio $\|A\|_\alpha / \|A\|_\beta$.

$\|\cdot\|_\alpha$ $\|\cdot\|_\beta$	$\|\cdot\|_1$	$\|\cdot\|_2$	$\|\cdot\|_\infty$	$\|\cdot\|_F$
$\|\cdot\|_1$	1	$\sqrt{n}$	n	$\sqrt{n}$
$\|\cdot\|_2$	$\sqrt{n}$	1	$\sqrt{n}$	1
$\|\cdot\|_\infty$	n	$\sqrt{n}$	1	$\sqrt{n}$
$\|\cdot\|_F$	$\sqrt{n}$	$\sqrt{n}$	$\sqrt{n}$	1

Table 2.1 Relationship Between Commonly Used Norms

2.3 An Iterative Improvement Scheme for Matrix Inversion

Let A be an n x n nonsingular matrix whose inverse is to be computed. Let $\mathbf{R}_k$ be the approximate inverse in the kth iteration. The residual matrix is a measure of deviation of the computed inverse from the actual inverse of the given matrix A. The residual matrix, $\mathbf{E}_k$, is given by

$$E_k = I - A R_k \tag{2.7}$$

The matrix $\mathbf{E}_k$ is also called the residual matrix. If $\mathbf{R}_0$ is the initial inverse matrix, then the inverse of A can be expressed in terms of the initial inverse, $\mathbf{R}_0$ and the initial residual matrix $\mathbf{E}_0$ as

$$A^{-1} = (R_0 R_0^{-1}) A^{-1} = R_0 (A R_0)^{-1} = R_0 (I - E_0)^{-1} = R_0 \sum_{i=0}^{\infty} (E_0)^i \tag{2.8}$$

The above expression is a power series in $\mathbf{E}_0$. The expression converges to $\mathbf{A}^{-1}$, the inverse of A, if the power series in $\mathbf{E}_0$ converges i.e if $\mathbf{E}_0$ is a convergent matrix. Hence the initial inverse calculated must be such that the residual matrix resulting from it must be convergent. Otherwise, the series begins to diverge and the iterative procedure will not be able to arrive at an inverse and the procedure becomes unstable. The expression used by Pan and Rief [14, 29] to calculate the inverse is given by

$$R_k = R_{k-1} (2I - A R_{k-1}) \tag{2.9}$$

where $\mathbf{R}_k$ is the inverse of A, obtained in the kth iteration. The following analysis shows that the method used by Pan and Reif [14, 29] is in fact the infinite series expansion given in (2.8) with only the first two terms taken into account.

The residual matrix in the computed initial inverse $\mathbf{R}_k$ is given by (2.7). Hence,

$$A R_k = I - E_k \tag{2.10}$$

$$(A\ R_k)^{-1} = (I - E_k)^{-1} \tag{2.11}$$

$$A^{-1} = R_k\,(I - E_k)^{-1} \tag{2.12}$$

$$A^{-1} = R_k\left(I + E_k + E_k^2 + \ldots\right) \tag{2.13}$$

By limiting the infinite series to the first two terms, we have,

$$A^{-1} = R_k\,(I + E_k) \tag{2.14}$$

$$A^{-1} = R_k\,(I + I - A\,R_k) \tag{2.15}$$

$$A^{-1} = R_k\,(2\,I - A\,R_k) \tag{2.16}$$

Alternatively, this can be written as,

$$R_k = R_{k-1}(2\,I - AR_{k-1}) \tag{2.17}$$

where, $\mathbf{R_k}$ is the inverse in the next iteration. This expression for the inverse is the same as given in (2.9).

It has been pointed out by Isaacsson and Keller [15] that the inclusion of the first three terms gives the fastest convergence of the iterative process for finding the inverse.

In the next section, it is analytically established that the fastest convergence can be achieved by including only the first three terms of the power series in the expression for the desired inverse given in (2.8).

2.4 Optimum Choice for the Number of Terms from the Infinite Series

It is evident that the expression for A^{-1} given in (2.8) is an infinite series. Considering a finite number of terms, say p, the resulting approximate inverse from (2.8) in the first iteration is, say $\mathbf{R_1}$, and the corresponding residual matrix is $\mathbf{E_1}$. Using $\mathbf{R_1}$ and $\mathbf{E_1}$ in (2.8), the approximate inverse $\mathbf{R_1}$ can be further improved to $\mathbf{R_2}$ during the second iteration. Successive application of this approach will result in further improvement of the inverse, reducing its deviation from the actual inverse. If either the residual matrix is zero or when the difference between the actual and the computed inverse is zero, then the inverse obtained is the actual inverse of the matrix A. In engineering, it is a standard practice to stop the computations as soon as the residual matrix assumes some small value in the neighbourhood of zero. Thus the iterative expression for finding the inverse in this manner can be written as

$$R_k = R_{k-1} \ (I + E_{k-1} + E_{k-1}^2 + \ldots\ldots + E_{k-1}^p) \tag{2.18}$$

$$E_k = I - A R_k \tag{2.19}$$

where $\mathbf{R_k}$ and $\mathbf{E_k}$ are the inverse and the residual matrices , respectively, in the kth iteration. Using (2.18) and (2.19), the residual matrix in the kth iteration can be written

as

$$\begin{aligned} E_k &= I - A R_{k-1} \left(I + E_{k-1} + \ldots\ldots + E_{k-1}^{p-1} \right) \\ &= I - \left(I - E_{k-1} \right) \left(I + E_{k-1} + \ldots\ldots + E_{k-1}^{p-1} \right) \\ &= E_{k-1}^{p} \end{aligned} \tag{2.20}$$

Substituting the value of E_{k-1}, E_{k-2}, E_{k-3}, etc., we have,

$$E_k = \left[E_{k-m} \right]^{p^m} \qquad m = 1,2,3,\ldots,k \tag{2.21}$$

For m = k, we have

$$E_k = E_0^{\,p^k} \tag{2.22}$$

From this generalised expression for the residual matrix, optimal number of terms from the infinite series given in (2.8) can be obtained. This is done next.

Since the residual matrix E_0 is assumed to be a convergent matrix, its eigenvalues $\lambda_1, \lambda_2, .., \lambda_n$ satisfy the condition

$$|\lambda_i| < 1 \tag{2.23}$$

Let

$$\lambda_s = \max_i |\lambda_i| \tag{2.24}$$

Thus, λ_s is the spectral radius of the residual matrix $\mathbf{E_0}$. The eigen values of the powers of the residual matrix $E_0^{p^k}$ are $\lambda_i^{p^k}$ [15]. Hence the spectral radius of $E_0^{p^k}$ is $\lambda_s^{p^k}$. The best choice of p is that value of p for which the least amount of computation yields an approximate inverse of desired accuracy, i.e., the best scheme could be defined as the one which yields the most accurate inverse for a given amount of computations.

It can be inferred from equations (2.18) and (2.19) that, to calculate the inverse $\mathbf{R_k}$ for a given matrix **A**, in the kth iteration, the residual matrix $\mathbf{E_{k-1}}$ and the inverse $\mathbf{R_{k-1}}$ are needed. The total number of operations required to compute $\mathbf{E_{k-1}}$ in (2.19) is $2n^3$, since calculating the product $\mathbf{AR_{k-1}}$ involves n^3 multiplications and $n^2(n-1)$ additions, and n^2 subtractions are needed to subtract the product $\mathbf{AR_{k-1}}$ from the identity matrix **I**. Equation (2.18) can be re-written as

$$R_k = R_{k-1} + R_{k-1}E_{k-1} + R_{k-1}E_{k-1}^2 + \ldots + R_{k-1}E_{k-1}^{p-1}) \qquad (2.25)$$

The successive terms in the above equation can be expressed as the product of the previous term and $\mathbf{E_{k-1}}$ as shown below:

$$R_k = R_{k-1} + R_{k-1}E_{k-1} + [R_{k-1}E_{k-1}]E_{k-1} + [R_{k-1}E_{k-1}^2]E_{k-1} + \ldots + [R_{k-1}E_{k-1}^{p-2}]E_{k-1} \qquad (2.26)$$

From this expression, it can be inferred that to calculate each term only one matrix multiplication is needed. Therefore, calculating matrix multiplication in (2.26) requires $(2n^3 - n^2)$ operations. Since there are (p-1) terms that need to be computed in (2.26), the

total number of operations to compute the (p-1) terms is $(p-1)(2n^3 - n^2)$. Further, (p-1) matrix additions of the terms are needed to evaluate the inverse matrix $\mathbf{R}_k$, which requires $(p-1)n^2$ operations. Moreover, in each iteration, calculating the residual matrix using (2.19) from the previous iteration using (2.19) involves $2n^3$ operations. Hence the total number of operations to calculate the inverse in each iteration is $2pn^3$. For k iterations the number of operations needed is

$$op.\ count = 2kpn^3 \tag{2.27}$$

If M operations are allowed, then the number of iterations is given by

$$k = \frac{M}{2pn^3} \tag{2.28}$$

The spectral radius of $\boldsymbol{E}_0^{p^k}$ is given by

$$\lambda_s\left(p^k\right) = \lambda_s\left(p^{\left[\frac{M}{2pn^3}\right]}\right) \tag{2.29}$$

To determine the value of p which results in a spectral radius of the residual matrix with a minimum value, the minimum value of $\lambda_s\left(p^{\frac{k}{p}}\right)$, where $k = M/n^3$, must be determined, since n and M are independent of p. Let

$$X = \lambda_s^{\left(p^{\frac{k}{p}}\right)} = \lambda_s^{Q} \tag{2.30}$$

$$Q = p^{\frac{k}{p}} \tag{2.31}$$

From equation (2.30) we have,

$$\ln Q = \frac{k}{p} \ln p \tag{2.32}$$

Differentiating (2.32) yields

$$\frac{1}{Q}\, dQ = -\frac{k}{p^2} \ln p \; dp + \frac{k}{p^2}\, dp \tag{2.33}$$

Rearranging the terms results in

$$\frac{dQ}{dp} = k \frac{Q}{p^2} \left[1 - \ln p\right] \tag{2.34}$$

But,

$$\frac{dX}{dp} = \frac{dX}{dQ} \frac{dQ}{dp} \tag{2.35}$$

Substituting the values from (2.34) into (2.35) and taking the differential of (2.32), we obtain,

$$\frac{dX}{dp} = Q\, \lambda_s^{(Q-1)} \frac{Q}{p^2}\, k \left[1 - \ln p\right] \tag{2.36}$$

For the spectral radius of $E_0^{p^t}$ to be a minimum, the differential $\frac{dX}{dp} = 0$. Hence

$$Q\,\lambda_s^{(Q-1)}\,\frac{Q}{p^2}\,[\,1 - \ln p\,] = 0 \tag{2.37}$$

Equation (2.37) implies the following three possibilities.

Case (i) : $\frac{1}{p^2} = 0$

This implies that $p = \infty$. This is not possible, since p has been assumed to be finite.

Case(ii): $Q^2\,\lambda_s^{(Q-1)} = 0$ (2.38)

This implies that

$$Q = 0 \quad or \quad \lambda_s = 0 \tag{2.39}$$

But, these conditions are not valid.

Case (iii): $1 - \ln p = 0$

This implies that

$$p = e \tag{2.40}$$

The value for e (the Napier's Constant) can be approximated to 2.718, hence, $p \approx 2.718$. The spectral radius λ_s, of the initial residual matrix E_0 is less than 1 since the matrix E_0 is convergent. Hence, the minimum value for the function $\lambda_s^{\left(p^{\frac{k}{p}}\right)}$ is obtained when $p^{1/p}$ is maximum. The solution for (2.37) identifies the critical point for the function in (2.29) at which the function may be either at its maximum or minimum value. The function will be maximum if λ_s is greater than 1. Since λ_s is less than 1 in the present case, the function will be minimum and the second derivative of $\lambda_s^{\left(p^{\frac{k}{p}}\right)}$ will be positive. Equation (2.33) is differentiated with respect to p using a symbolic manipulation software (MAPLE). The output from this is shown in Appendix A. The simplified form of the second derivative is given by

$$\frac{d^2X}{dp^2} = \frac{\lambda_s^{p^{\frac{1}{p}}} \ln\lambda_s}{p^2}\left((1-\ln p)^2\left[\ln\lambda_s\, p^{-2\frac{p-1}{p}} + p^{-\frac{2p-1}{p}}\right] - p^{\frac{5p-1}{p}}(3+2\ln p)\right) \quad (2.41)$$

Substituting p = e in (2.41), the second derivative is given by

$$\frac{d^2X}{dp^2} = 0.135\ \lambda_s^{1.445}\ \ln(\lambda_s)\left(10^{-10}\ \ln(\lambda_s) - 0.532\right) \quad (2.42)$$

The above equation is always positive for a convergent residual matrix since λ_s is less than 1. Hence, it can be concluded that the second derivative is positive.

From the above discussion, it can be concluded that p = 3 indeed yields the minimum value of the spectral radius of residual matrix in the kth iteration. Since k has to be an integer, we can examine the values of the function in (2.30) for integer values The spectral radius of E_k given in (2.30) is a function that monotonically decreases from p = 0 to p = e and monotonically increases for any value of p greater than e. Hence, it is sufficient if we examine the value of the function in (2.30) for the integer values p = 2 and p = 3. The value of the function in (2.30) at p = 3 is lower than the value at p = 2. Hence, the optimum choice of p is 3.

Fixing the amount of computations indirectly implies that if the computational time allowed to carry out the inverse calculations is fixed, best results can be obtained by choosing no more than the first three terms from the infinite series in (2.18). Including more than three terms in the infinite series results in more accurate inverse calculations. This is, however, achieved only at the expense of increased computations.

2.5 Iterative Inverse Formulation Using a Reduced Number of Operations

The iterative matrix inverse formulation given in (2.18) can be rewritten using the formula for the residual matrix given in (2.22). The inverse in successive iterations can be re-written as

$$R_1 = R_0\left[I + E_0 + E_0^2 + \ldots + E_0^{p-1}\right] \tag{2.43}$$

$$R_2 = R_1\left[I + E_0^p + \left(E_0^p\right)^2 + \left(E_0^p\right)^3 + \ldots + \left(E_0^p\right)^{p-1}\right] \tag{2.44}$$

$$R_3 = R_2\left[I + \left(E_0^{p^2}\right) + \left(E_0^{p^2}\right)^2 + \left(E_0^{p^2}\right)^3 + \ldots + \left(E_0^{p^2}\right)^{p-1}\right] \tag{2.45}$$

Thus, the general form is derived as

$$R_k = R_{k-1}\left[I + \left(E_0^{p^{k-1}}\right) + \left(E_0^{p^{k-1}}\right)^2 + \left(E_0^{p^{k-1}}\right)^3 + \ldots + \left(E_0^{p^{k-1}}\right)^{p-1}\right] \tag{2.46}$$

Repeated substitution for the value of $\mathbf{R}_{k-1}$ in (2.46) results in

$$\begin{aligned} R_k = R_0 &\left[I + E_0 + E_0^2 + \ldots + E_0^{p-1}\right]\left[I + E_0^{p} + \left(E_0^{p^2}\right)^2 + \ldots + \left(E_0^{p^2}\right)^{p-1}\right] \\ &\left[I + E_0^{p^{k-1}} + \left(E_0^{p^{k-1}}\right)^2 + \ldots + \left(E_0^{p^{k-1}}\right)^{p-1}\right] \\ = R_0 &\prod_{h=0}^{k-1}\left[I + \left(E_0^{p^h}\right) + \left(E_0^{p^h}\right)^2 + \ldots + \left(E_0^{p^h}\right)^{k-1}\right] \end{aligned} \tag{2.47}$$

The following derivation arrives at the number of operations involved in calculating the inverse.

Let the number of terms chosen from the infinite series in the inverse formulation given in (2.47) be p = 2. Then, the inverse in the first iteration is given by

$$R_1 = R_0\left[I + E_0\right] \tag{2.48}$$

The number of operations to multiply two matrices is $2n^3$-n^2 for a matrix of order nxn. The number of operations to add two matrices is n^2. Further, from the discussions presented in Section 2.4 for the optimum choice of p from the power series, it is noted that the number of operations to calculate the residual matrix $\mathbf{E}_0$ is $2n^3$. Hence the total

number of operations to calculate the inverse $\mathbf{R}_1$ is given by

$$\begin{aligned} N_1 &= (2n^3 - n^2) + n^2 + 2n^3 \\ &= 4n^3 \end{aligned} \tag{2.49}$$

The inverse for p = 2 in the 2nd iteration is given by

$$R_2 = R_1\left[I + E_0^2\right] \tag{2.50}$$

This calculation involves 2 matrix multiplications and 1 matrix addition. The number of operations to calculate $\mathbf{R}_2$ is

$$\begin{aligned} N_2 &= 2n^3 - n^2 + n^2 + 2n^3 - n^2 \\ &= 4n^3 - n^2 \end{aligned} \tag{2.51}$$

The inverse for p = 2 in the 3rd iteration is given by

$$\begin{aligned} R_3 &= R_2\left[I + E_0^4\right] \\ &= R_2\left[I + E_0^2 . E_0^2\right] \end{aligned} \tag{2.52}$$

Thus calculating $\mathbf{R}_3$ involves 2 matrix multiplications and 1 matrix addition. The number of operations to calculate $\mathbf{R}_3$ is also $4n^3$-n^2. Except for the first iteration, all other iterations have the same operation count equal to $4n^3$-n^2. Hence, if k iterations are performed, the total number of operations required is given by

$$N = (k-1)(4n^3 - n^2) + 4n^3$$
$$= 4n^3k - n^2(k-1) \tag{2.53}$$

Using three terms from the infinite series, the inverse in the 1st iteration is given by

$$R_1 = R_0\left[I + E_0 + E_0^2\right] \tag{2.54}$$

The calculation of $\mathbf{R}_1$ involves three matrix multiplications and 2 matrix additions. The operations count to calculate the inverse $\mathbf{R}_1$ is

$$N_1 = 2n^3 - n^2 + n^2 + 2n^3 + n^2 + 2n^3 - n^2$$
$$= 6n^3 \tag{2.55}$$

The inverse $\mathbf{R}_2$ for p = 3 in the 2nd iteration can be written as

$$R_2 = R_1\left[I + E_0^3 + \left(E_0^3\right)\left(E_0^3\right)\right] \tag{2.56}$$

The operational count N_2 is equal to

$$N_2 = 2n^3 - n^2 + n^2 + 2n^3 - n^2 + n^2 + 2n^3 - n^2$$
$$= 6n^3 - n^2 \tag{2.57}$$

Similarly, the inverse $\mathbf{R}_3$ is given by

$$R_3 = R_2\left[I + E_0^9 + E_0^{18}\right]$$
$$= R_2\left[I + \left(E_0^6\right)\left(E_0^3\right) + \left(E_0^9\right)\left(E_0^9\right)\right] \tag{2.58}$$

The operation count N_3 is given by

$$\begin{aligned} N_3 &= 2n^3 - n^2 + n^2 + 2n^3 - n^2 + n^2 + 2n^3 - n^2 \\ &= 6n^3 - n^2 \end{aligned} \tag{2.59}$$

To perform k iterations, the total number of operations using p = 3 is

$$\begin{aligned} N &= (k-1)(6n^3 - n^2) + 6n^3 \\ &= 6n^3 k - n^2(k-1) \end{aligned} \tag{2.60}$$

For p = 4, the inverse in the first iteration using the iterative improvement scheme is given by

$$\begin{aligned} R_1 &= R_0\left[I + E_0 + E_0^2 + E_0^3\right] \\ &= R_0\left[I + E_0 + E_0^2 + E_0\left(E_0^2\right)\right] \end{aligned} \tag{2.61}$$

The number of operations N_1 is given by

$$\begin{aligned} N_1 &= 2n^3 - n^2 + n^2 + 2n^3 + n^2 + 2n^3 - n^2 + n^2 + 2n^3 - n^2 \\ &= 8n^3 \end{aligned} \tag{2.62}$$

The inverse in the second iteration is given by

$$\begin{aligned} R_2 &= R_1\left[I + E_0^4 + E_0^8 + E_0^{12}\right] \\ &= R_1\left[I + E_0^2 . E_0^2 + E_0^4 . E_0^4 + E_0^4 . E_0^8\right] \end{aligned} \tag{2.63}$$

The operation count N_2 for R_2 is then calculated as

$$\begin{aligned} N_2 &= 2n^3 - n^2 + n^2 + 2n^3 - n^2 + n^2 + 2n^3 - n^2 + n^2 + 2n^3 - n^2 \\ &= 8n^3 - n^2 \end{aligned} \tag{2.64}$$

The inverse in the third iteration is

$$\begin{aligned} R_3 &= R_2\left[I + E_0^{16} + E_0^{32} + E_0^{48}\right] \\ &= R_2\left[I + E_0^{8} \cdot E_0^{8} + E_0^{16} \cdot E_0^{16} + E_0^{32} \cdot E_0^{16}\right] \end{aligned} \tag{2.65}$$

The operation count N_2 is computed as

$$\begin{aligned} N_2 &= 2n^3 - n^2 + n^2 + 2n^3 - n^2 + n^2 + 2n^3 - n^2 + n^2 + 2n^3 - n^2 \\ &= 8n^3 - n^2 \end{aligned} \tag{2.66}$$

Thus the total operational count for k iterations for p = 4 is given by

$$\begin{aligned} N &= (k-1)(8n^3 - n^2) + 8n^3 \\ &= 8n^3k - n^2(k-1) \end{aligned} \tag{2.67}$$

From equations (2.62), (2.64) and (2.66), it can be inferred that the number of operations for a given value of p for k iterations is obtained as

$$N = 2n^3kp - n^2(k-1) \tag{2.68}$$

The expression for the number of iterations obtained in Section (2.4) for the number of operations to calculate the inverse using the general iterative scheme is n^3pk. Thus, there

is a reduction in the number of operations needed to compute the inverse using the simplified generalized formulation presented in (2.47). The number of operations needed is reduced by a factor $(k-1)n^2$. Section 2.4 shows the optimum choice of terms from the power series is 3 for a fixed amount of operations. Hence, using p =3 in (2.47), results in

$$R_k = R_0 \prod_{h=0}^{k-1} \left[I + \left(E_0^{p^h} \right) + \left(E_0^{p^h} \right)^2 \right] \tag{2.69}$$

The expression presented in (2.69) has the advantage that the calculation of the inverse $\mathbf{R}_k$ in one iteration, involves the use of the inverse obtained in the previous iteration along with the computation of only 2 new terms involving the initial residual matrices. The calculation of these terms involves the multiplication of the error matrices obtained previously. For example, the first iteration involves calculating $\mathbf{E}_0$ and E_0^2. For the next iteration, we need to calculate E_0^3, which can be obtained as a product of $\mathbf{E}_0$ and E_0^2, and E_0^6, which can be obtained as the square of E_0^3. The matrices I, E_0^3 and E_0^6 are added, and the result is multiplied by $\mathbf{R}_1$ to obtain $\mathbf{R}_2$. Similarly, the matrices needed for the third iteration is E_0^9 and E_0^{18}. The term E_0^9 can be obtained as the product of E_0^3 and E_0^6, and E_0^{18} can be computed as the square of E_0^9. The matrices I, E_0^9 and E_0^{18} are added, and the result is multiplied by $\mathbf{R}_2$ to obtain $\mathbf{R}_3$. Thus, calculating E_0^{3h} and

$\left(E_0^{3h}\right)^2$ in (2.69) requires only a single matrix multiplication.

The problem of finding the inverse of a matrix has been effectively reduced to one of performing matrix multiplications. There are several known parallel algorithms for performing matrix multiplications which can be used. Using fast parallel algorithms for matrix multiplications provides asymptotic convergence for obtaining the inverse of a matrix [21, 30]. The matrix multiplication algorithms by Strassen [22], Winograd [30] or some bilinear algorithms [31] can also be used.

The successful implementation of this method or any iterative matrix inversion scheme lies in the proper choice of the initial inverse. If the initial inverse, $\mathbf{R}_0$, used is such that successive powers of the residual matrix $\mathbf{E}_0$, begin to diverge then, the iterative scheme will fail to arrive at an inverse, and will become unstable. The following section describes the initial inverse formulation developed in this study.

2.6 Initial Inverse Formulation

An inverse matrix $\mathbf{R}_0$, is considered a good initial approximation of the inverse of a matrix **A** if the residual matrix $\mathbf{E}_0$ resulting from it is convergent. The residual matrix $\mathbf{E}_0$ is given by

$$E_0 = I - AR_0 \tag{2.70}$$

Hence,

$$(AR_0)^{-1} = (I - E_0)^{-1} \tag{2.71}$$

Since

$$I = (I - E_0)(I - E_0)^{-1} \tag{2.72}$$

We have

$$(I - E_0)^{-1} = I + E_0(I - E_0)^{-1} \tag{2.73}$$

With respect to some matrix norm,

$$\|(I - E_0)^{-1}\| \leq 1 + \|E_0\| \|(I - E_0)^{-1}\| \tag{2.74}$$

Dividing both sides by $\| (I - E_0)^{-1} \|$, we have

$$1 \leq \frac{1}{\|(I - E_0)^{-1}\|} + \|E_0\| \tag{2.75}$$

From (2.70) we have

$$R_0^{-1} A^{-1} = (I - E_0)^{-1} \tag{2.76}$$

Hence,

$$A^{-1} = R_0 (I - E_0)^{-1} \tag{2.77}$$

With respect to some matrix norm, we have

$$\begin{aligned} \|A^{-1}\| - \|R_0 (I - E_0)^{-1}\| \\ \leq \|R_0\| \, \|(I - E_0)^{-1}\| \end{aligned} \tag{2.78}$$

Using (2.75) in (2.78), we have,

$$\|A^{-1}\| \leq \|R_0\| \, \|(I - E_0)^{-1}\| \leq \frac{\|R_0\|}{1 - \|E_0\|} \tag{2.79}$$

Since

$$\|(I - E_0)^{-1}\| \leq \frac{1}{1 - \|E_0\|} \tag{2.80}$$

From (2.79), we have,

$$1 - \|E_0\| \leq \frac{\|R_0\|}{\|A^{-1}\|} \tag{2.81}$$

Thus,

$$c(1 - \|E_0\|) = \frac{\|R_0\|}{\|A^{-1}\|} \qquad c \geq 1 \tag{2.82}$$

Hence the norm of the residual matrix can be given by

$$\|E_0\| = 1 - c' \frac{\|R_0\|}{\|A^{-1}\|} \qquad c' = \frac{1}{c} \leq 1 \tag{2.83}$$

However, the residual matrix $\mathbf{E}_0$ must be convergent for the initial inverse approximation, $\mathbf{R}_0$ to be valid. If this condition has to be satisfied, we have to have

$$c' \frac{\|R_0\|}{\|A^{-1}\|} \leq 1 \tag{2.84}$$

Let the initial inverse $\mathbf{R_0}$ be given by

$$R_0 = \alpha A \tag{2.85}$$

Case (i) : Infinity Norm

Let the value of α in (2.85) be given by

$$\alpha = \frac{1}{\|A\|_\infty^2} \tag{2.86}$$

Thus,

$$\frac{\|R_0\|}{\|A^{-1}\|} = \frac{\|\alpha A\|}{\|A^{-1}\|} = \frac{\|A\|}{\|A\|_\infty^2 \|A^{-1}\|} \tag{2.87}$$

Using the infinity norm, (2.87) can be written as

$$\frac{\|R_0\|_\infty}{\|A^{-1}\|_\infty} = \frac{\|A\|_\infty}{\|A\|_\infty^2 \|A^{-1}\|_\infty} \tag{2.88}$$

The condition number of A with respect to a matrix norm is given by

$$K(A) = \|A^{-1}\| \, \|A\| \tag{2.89}$$

The condition number of A with respect to a matrix norm determines if small changes in the given matrix A results in small changes in the inverse calculation. A matrix is said

to be well conditioned if the condition number is small, and it is ill-conditioned otherwise. Further we know that the condition number of any matrix is greater than or equal to 1, since

$$\|A\|\|A^{-1}\| \geq \|I\| = 1 \tag{2.90}$$

Using (2.89) in (2.88), we have

$$\frac{\|R_0\|_\infty}{\|A^{-1}\|_\infty} = \frac{1}{K_\infty(A)} \leq 1 \tag{2.91}$$

where $K_\infty(A)$ is the condition number of A with respect to the infinity norm. Since $0 < c' \leq 1$, (2.84) is satisfied. This verifies that the initial inverse formulation using the infinity norm is correct.

Case (ii) : One Norm

Let the value of α in (2.85) be given by

$$\alpha = \frac{1}{\|A\|_1^2} \tag{2.92}$$

Substituting (2.92) in (2.84), we have,

$$\frac{\|R_0\|}{\|A^{-1}\|} = \frac{\|\alpha A\|}{\|A^{-1}\|} = \frac{\|A\|}{\|A\|_1^2\|A^{-1}\|} \tag{2.93}$$

Using the one norm, (2.93) can be represented as

$$\frac{\|R_0\|_1}{\|A^{-1}\|_1} = \frac{\|A\|_1}{\|A\|_1^2 \|A^{-1}\|_1} \tag{2.94}$$

Equation (2.94) can be re-written as

$$\frac{\|R_0\|_1}{\|A^{-1}\|_1} = \frac{1}{K_1(A)} \leq 1 \tag{2.95}$$

where $K_1(A)$ is the condition number of A with respect to the one norm. Since $0 < c' \leq 1$, (2.84) is satisfied. This verifies the correctness of the initial inverse formulation based on the one norm.

Case (iii) : Frobenius Norm

Let the value of α in (2.85) be given by

$$\alpha = \frac{1}{\|A\|_F^2} \tag{2.96}$$

Substituting (2.96) in (2.84), we have

$$\frac{\|R_0\|}{\|A^{-1}\|} = \frac{\|\alpha A\|}{\|A^{-1}\|} = \frac{\|A\|}{\|A\|_F^2 \|A^{-1}\|} \tag{2.97}$$

Using the Frobenius norm, (2.97) can be written as

$$\frac{\|R_0\|_F}{\|A^{-1}\|_F} - \frac{\|A\|_F}{\|A\|_F^2 \|A^{-1}\|_F} \tag{2.98}$$

Using (2.98) in (2.85), we have

$$\frac{\|R_0\|_F}{\|A^{-1}\|_F} - \frac{1}{K_F(A)} \leq 1 \tag{2.99}$$

where $K_F(A)$ is the condition number of A with respect to the Frobenius norm. Since $0 < c' \leq 1$, (2.85) is satisfied. Thus, the correctness of the initial inverse formulation using the Frobenius norm is verified.

Case (iv) : Combined Norm

One another choice of the initial inverse is given by

$$R_0 = \alpha A^T \tag{2.100}$$

where α is given by

$$\alpha = \frac{1}{n \|A\|_\infty \|A\|_1} \tag{2.101}$$

Substituting α in (2.100), we have

$$\frac{\|R_0\|}{\|A^{-1}\|} - \frac{\|A^T\|}{n \|A\|_\infty \|A\|_1 \|A^{-1}\|} \tag{2.102}$$

With respect to the infinity norm,

$$\frac{\|R_0\|_\infty}{\|A^{-1}\|_\infty} = \frac{\|A^T\|_\infty}{n\,\|A\|_\infty \|A\|_1 \|A^{-1}\|_\infty} \tag{2.103}$$

Using Table 2.1, we get

$$\frac{\|R_0\|}{\|A^{-1}\|} = \frac{1}{K_\infty(A)} \leq 1 \tag{2.104}$$

This form is same as the one shown in equation (2.91) for case (i), formulation is valid. However, it should be noted that the convergence obtainable using this formulation for choosing the initial inverse, is affected by the order of the matrix to be inverted.

2.7 Stopping Criterion

The iterative procedures require a stopping criterion to limit the number of iterations carried out to obtain the desired results. A stopping criterion is derived for this purpose. The norm of residual matrix in the kth iteration is given by

$$\| E_k \| = \|E_0^{3^k}\| \tag{2.105}$$

Using initial inverse formulation presented in Section 2.6 for the iterative procedure results in a convergent residual matrix, i.e.

$$\|E_0\| = q, \qquad \textit{where} \qquad q < 1 \tag{2.106}$$

Then, by the properties of norms, (2.105) becomes

$$\begin{aligned} \| E_k \| &\leq \| E_0 \|^{3^k} \\ &\leq q^{3^k} \\ &< 1 \end{aligned} \tag{2.107}$$

The deviation in the computed inverse from the actual inverse is given by

$$A^{-1} - R_k = A^{-1}(I - AR_k) \tag{2.108}$$

With respect to some matrix norm, we have,

$$\begin{aligned} \| A^{-1} - R_k \| &\leq \| A^{-1} \| \| (I - AR_k) \| \\ &\leq \| A^{-1} \| \| E_k \| \end{aligned} \tag{2.109}$$

Hence,

$$\frac{\| A^{-1} - R_k \|}{\| A^{-1} \|} \leq \| E_k \| \tag{2.110}$$

From (2.107) and (2.110), we have,

$$\frac{\| A^{-1} - R_k \|}{\| A^{-1} \|} \leq q^{3^k} \tag{2.111}$$

Once the initial inverse is formulated, the residual matrix E_0 is known. Hence, the value of q, which is the norm of E_0 can be determined. Since the value of p, i.e. the number of terms in the infinite series is also known, substituting these values in equation

(2.111), the number of iterations needed can be determined, when the allowable percentage deviation in the computed inverse is given.

2.8 Summary

In this chapter, an efficient iterative scheme for the inversion of matrices which uses less number of operations to arrive at the desired inverse compared to the conventional approach has been presented. It has been shown that the number of operations needed to calculate the inverse is reduced by $n^2(k-1)$ compared to the conventional approach. The efficiency of the inversion scheme is further improved by choosing an optimum number of terms from the infinite series used in the iterative formulation, for a fixed number of operations. An analytical proof has been derived for this purpose, and the optimum value for p has been established to be *e*, the Napier's constant. Since the spectral radius function monotonically decreases upto *e* and increases thereafter, integer values of p in the neighbourhood of *e* have been examined. An investigation of the nature of the function for the spectral radius reveals that the optimum integer value for p is 3. The generalized iterative formulation has been further improved by taking only the first three terms. Iterative schemes require an initial approximation that is iteratively improved, and also a stopping criterion. Formulations to choose the initial inverse using the one, infinity and Frobenius norms and a stopping criterion have been derived. From the formulation for the stopping criterion, factors that affect the stopping criterion have been identified.

CHAPTER 3

EXPERIMENTAL RESULTS

3.1 Introduction

This chapter presents the experimental verification of the theoretical results obtained in Chapter 2. To carry out the tests, implementation is carried out using parallel processing hardware and software, namely, the T-800 transputer and Occam.

Section 3.2 summarizes some of the features of transputers. Section 3.3 presents the structure of the Occam procedures developed for implementation in this thesis. The results of various experiments carried out are detailed in Section 3.4. Tests are carried out for different orders of matrices (10x10, 25x25 and 100x100) to illustrate that the method can be applied for high order matrices without a drastic increase in the number of iterations. Further, test results for different choices of initial inverse using the one, infinity and Frobenius norms for p = 2, 3 and 4 are included. To demonstrate that for a fixed amount of computations, the error is minimum for p = 3, tests have been carried out and are discussed in this chapter.

3.2 Hardware and Software Platform

Transputers can be used in applications ranging from embedded systems to supercomputers. Many computationally intensive applications such as image synthesis,

digital signal processing, robot control etc., can use large arrays of transputers for carrying out parallel computations. The system performance depends on the number of transputers, the speed of inter-transputer communication, and the performance of each transputer.

The word transputer is a contraction of the words transistor and computer. It is a new generation VLSI architecture which explicitly supports parallelism. The INMOS transputer consists of a high-performance processor, on-chip RAM, and inter-processor links, all on a single chip of silicon [32]. The T-800 transputer is a 32 bit 10MIPS RISC processor with 4KB of local RAM, an external memory interface and four link interface units.

The on-chip RAM allows fast data access and fast code execution in comparison to off-chip performance. By holding frequently accessed data in local RAM, a high level of performance is obtained by the stack oriented six-register architecture. The registers A, B and C form the evaluation stack. The workspace register points to an area of store where local variables for currently executing processes are stored. A process can be in any one of the states: suspended, executing or executable. A process that is suspended is waiting for some input or output action to be completed on a channel. All the processes that are capable of being executed are kept on the dispatch queue. The scheduler is a hardware facility which moves from one executing process to another very quickly.

The interprocessor links are autonomous DMA engines and permit any number of transputers to be connected together in arbitrary networks providing extra processing power. Interprocessor communication is supported through these links. They avoid the communication bottleneck of a single bus system. Bit-serial communication rates of 20 Mbits/s are achieved via the four bidirectional interprocessor links. Serial to parallel data conversion is performed by the on-chip link interface hardware. Communication between the transputers must be via explicitly passing the data along the links. Hence, if all the processors need a data set, they must each have a copy of it. If the data set gets updated, arrangements must be made for keeping all the copies of that data set up to date. This overhead means that it pays to send the processors only the data that they logically need. Hence, array handling programs must be designed so that individual processors need only part of the array, or only part of it at a time. The absence of shared memory affects coding techniques because most languages implicitly assume the presence of shared memory. One of the major challenges of using transputer networks is splitting up the application over multiple transputers and meeting the performance requirements of the application while making effective use of the transputers available. This is because, interprocessor communication is only through the physical serial communication links.

The external memory interface allows linear access to a total memory of 4 gigabytes in the 32-bit T-800 transputers. Transputers are hardware processors which execute software processes. It has the feature of multi-tasking in a single processor and multi-threading any number of processes in a single or multiple transputers. Multi-tasking

allows many tasks to share data using a common local memory of 4Kbytes on a single chip. Though transputers allow multi-tasking, the programming language used in the implementation has to support this feature. Whereas, the multi-threading on a single transputer simulates the code execution on multiple transputers. Multi-threading allows data sharing through explicitly defined communication channels instead of using the common shared memory. These channels are software channels if the implementation is carried out on a single transputer. When implemented on multiple transputers, these channels are the hardware channels. This distinction needs to be defined only in a configuration file which places the channels on hardware or software links. The configuration file also contains the information on the number of transputers used in the physical system.

The programming model for transputers is based on the Occam model of concurrency using message passing for multi-threading, Occam offers the best support for utilizing the concurrency and communication facilities offered by transputers [33]. Processes are connected together using synchronized, unbuffered, point-to-point, uni-directional communication channels. Occam does not support shared variables for two or more processes executing on a single transputer to share data, i.e. multi-tasking is not supported by Occam. Hence, message passing is the only form of communication between parallel processes. Thus, the same model of concurrency is used both within processes and between processes. This has two advantages: it prevents a class of programming errors associated with shared variable communication and it makes it easier

to change the way that a particular program is split-up between processes. To indicate that two, or more processes are concurrent (multi-threading), the *PAR* construct is provided in Occam. All subprocesses of a *PAR* must either be completely independent of each other or interact using channels and a synchronous communication. From program development standpoint, the *PAR* construct can be used to express concurrency. These features make Occam a suitable choice for programming on transputers.

3.3 Algorithm Structure

This section presents the structure of the procedures developed using Occam to implement the scheme proposed in Chapter 2 to compute the inverse of a matrix. The computations that are capable for parallel execution are identified and implemented. The algorithms developed for implementation are *NORM, TRANSPOSE, INITIAL INVERSE, MATMULT* and *INVITER* as presented below.

Procedure NORM

Input: An n x n matrix A

Output: The infinity norm of A

```
for i= 0 to n-1 dopar
  n[i] = 0
odpar
for i = 0 to n-1 dopar
  for j = 0 to n-1
```

```
        n[i] = abs(A[i][j]) + n[i]
    odpar
```

Thus, the sum of all the rows or columns of A can be carried out in parallel. From the calculated values of norm, the maximum is determined as the infinity norm. The same procedure is used for the other norms discussed in Chapter 2 to obtain the desired inverse.

Procedure TRANSPOSE

Input: An n x n matrix A

Output: The transpose of A

```
for i= 0 to n-1 dopar
        for j= 0 to n-1 dopar
                A^t[i,j] = A[j,i]
odpar
```

The transpose of the matrix is needed to calculate the initial inverse if the Frobenius norm is used.

Procedure INITIAL INVERSE

Input: A, norm

Output: R_0

```
for i= 0 to n-1 dopar
    for j= 0 to n-1 dopar
        R0[i,j] = norm * A[i,j]
odpar
```

The procedures to calculate the inverse using the one norm and the Frobenius norm are similar to the one given above except for the fact that the norm and the initial inverse are calculated using (2.92) and (2.96) respectively.

Several parallel algorithms for parallel matrix multiplication are available. There are systolic algorithms for matrix multiplication discussed by Chaudhuri [34]. Another matrix multiplication algorithm that can be implemented in parallel is the Strassen's algorithm discussed in Chapter 1. As a matter of fact, a parallel implementation of this is discussed by Gianey [21]. One another algorithm which can be used is the commutative Winograd algorithm [29]. Francomano et al. [35] have discussed the comparison of different methods for matrix multiplication. However, a simple parallel implementation of the matrix multiplication algorithm as given below is used in this chapter as the focus here is on validating the concepts developed in Chapter 2.

Procedure MATMULT

Input: Matrices A, B

Output: Matrix C = AB

```
for j = 0 to n-1 dopar
        C[i][j] = 0
        for l = 0 for n-1 dopar
                C[i][j] = C[i][j] + A[i][l]*B[l][j]
        odpar
odpar
```

Procedure INVITER

Input : R_0, E_0

Output : A^{-1} = INVERSE

```
matmult(SQRE0, E0, E0)
matmult(CUBE0, SQRE0, E0)
for i = 0 to n-1 dopar
        for j = 0 for n-1 dopar
                INVERSE[i][j] = IDY[i][j] + E0[i][j] + SQRE0[i][j]
        odpar
odpar
dopar
        matmult(INVERSE1, R0, INVERSE)
```

```
        matmult(SQRE0, CUBE0, CUBE0)
    odpar

    for i = 0 to n-1 dopar
        for j = 0 for n-1 dopar
            INVERSE2[i][j] = IDY[i][j] + CUBE0[i][j] + SQRE0[i][j]
        odpar
    odpar
    matmult(INVERSE, INVERSE1, INVERSE2)
    for i = 0 to k-1
        matmult(E0, CUBE0, SQRE0)
        matmult(SQRE0, E0, E0)
        CUBE0:= E0
        for i = 0 to n-1
            for j = 0 for n-1 dopar
                INVERSE1[i][j] = IDY[i][j] + E0[i][j] + SQRE0[i][j]
            odpar
        matmult(INVERSE2, INVERSE, INVERSE1)
        INVERSE := INVERSE2
```

The initial error R_0 and the residual matrix E_0 are calculated using the algorithm matmult for matrix multiplications. The variables SQRE0 and CUBE0 are used in the first

step to store the products E_0^2 and E_0^3 respectively. The computations involving the addition of matrices to evaluate the inverse, $\mathbf{R}_1$, are performed in parallel and the result is stored in the variable INVERSE. The inverse $\mathbf{R}_1$ is computed as a product of the initial inverse input matrix R_0 and INVERSE, and the result is stored in the variable INVERSE1. The value E_0^6 is computed in parallel with INVERSE1 and the result is stored in SQRE0. The intermediate result of adding matrices IDY, CUBE0 and SQRE0 to compute R_2 is carried out in parallel, and the result is stored in INVERSE2. The inverse R_2, stored in INVERSE, is obtained by multiplying INVERSE1 and INVERSE2. The computations to find the inverse in subsequent iterations is same as those to calculate R_2.

3.4 Implementation Methodology

The INMOS T-800 transputer is connected to a host PC, to give access to the terminal and the filing system of the host computer. The processor intensive part of performing matrix manipulations is carried out on the transputer. The host computer stores the code to be executed in the transputer in its filing system, and boots up the transputer with the code when the application is required to run. Since the matrix inversion is done as a stand alone application in this case, the booting of the transputer and the program execution in the transputer is done at the command line prompt of the host PC. The main procedure downloads the matrix to be inverted from a file in the host to the transputer. This is done by the procedure *trial-load*. The transpose of the matrix

if needed, (if Frobenius norm is used to calculate the initial inverse) and the initial inverse are computed. The inverse calculations in each iteration are carried out by the procedure *inviter* which makes use of the procedures *matmult, norm, and initial inverse*. These procedures are described in Section 3.3. The matrix multiplication is done by the procedure *matmult*. The parallel operations within *matmult* are the vector load for initialization and vector multiply for the inner loop of additive multiplications.

3.5 Experimental Results

Experiments are carried out to test the various concepts developed in Chapter 2. For this purpose, several sets of matrices of orders ranging from 10 to 100 are randomly generated and used. Tests are carried out to verify the optimum choice of terms to calculate the inverse, to show that convergence is assured if the initial inverse is chosen according to the schemes developed in Chapter 2, and to show that the number of iterations needed for the calculation of the inverse is not sensitive to the order of the matrices.

In order to verify that for a fixed amount of computations, the most accurate inverse is obtained for p = 3, $\| E_k \|_\infty$ is plotted against the number of iterations, k. Tests are carried out with the initial inverse formulation using the infinity norm, for matrices of order 10x10, 25x25 and 100x100. Figure 3.1 shows the results obtained for a matrix of order 10x10. From Fig. 3.1, the value of the error $\| E_k \|_\infty$ is determined for a fixed

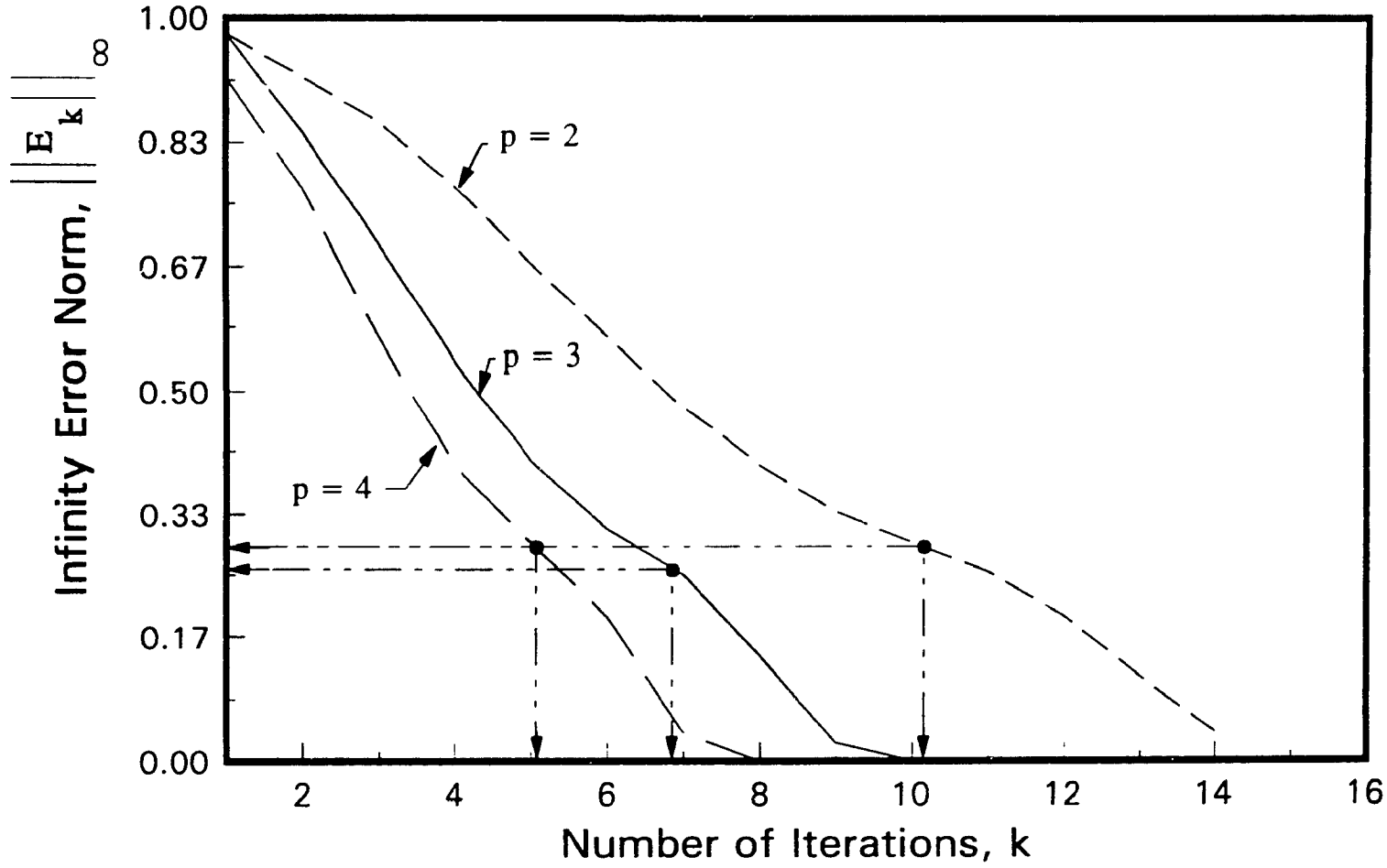

Fig. 3.1 Convergence for various values of p for initial inverse using infinity norm, n = 10x10

amount of computations, by arbitrarily fixing the number of computations, say, 8×10^7 operations, and using the formula

$$No\ of\ operations = 2n^3pk - (k-1)n^2 \tag{3.1}$$

derived in Chapter 2. Since the value of the number of operations is fixed, and since the values of p are 2, 3 and 4, the number of iterations needed in each case is obtained. From the knowledge of the number of iterations, the values of $\| E_k \|_\infty$ have been determined from Fig. 3.1. The procedure outlined above is repeated for another value of the number of operations, 4×10^6. The value of $\| E_k \|_\infty$ is once again obtained from Fig. 3.1. The number of iterations needed and the value of $\| E_k \|_\infty$ obtained in each case is tabulated in Table 3.1.

Total Number of Operations	p = 2		p = 3		p = 4	
	Iteration	$\| E_k \|_\infty$	Iteration	$\| E_k \|_\infty$	Iteration	$\| E_k \|_\infty$
5.5×10^4	14.1	0.043	9.3	0.015	6.95	0.043
5.0×10^4	10.2	0.29	6.8	0.26	5.1	0.29

Table 3.1 Number of Iterations Required and Error Norm for Fixed Amount of Operations for Various Values of p Using Infinity Norm, n = 10x10.

Figure 3.2 presents the results of the same test, as mentioned above, for a matrix of order 25x25. The number of operations used is fixed arbitrarily to $8x10^5$ and $6x10^5$. The values of $\| E_k \|_\infty$ is obtained for each case from Fig. 3.2, for p = 2, 3, and 4. The values of $\| E_k \|_\infty$ are presented in Table 3.2.

Total Number of Operations	p = 2		p = 3		p = 4	
	Iteration	$\| E_k \|_\infty$	Iteration	$\| E_k \|_\infty$	Iteration	$\| E_k \|_\infty$
8×10^5	12.9	0.16	8.6	0.12	6.4	0.16
6×10^5	9.7	0.38	6.4	0.34	4.8	0.38

Table 3.2 Number of Iterations Required and Error Norm for Fixed Amount of Operations for Various Values of p Using Infinity Norm, n = 25x25.

Figure 3.3 shows the results for a matrix of order 100x100. The results of the tests are summarized in Table 3.3.

Results of Tables 3.1-3.3 are also plotted in Fig. 3.4. For integer values of p, the values of $\| E_k \|_\infty$ increases below and above p= 3. The nature of $\| E_k \|_\infty$ has been described in Chapter 2 in the context of optimizing its spectral radius with respect to p.

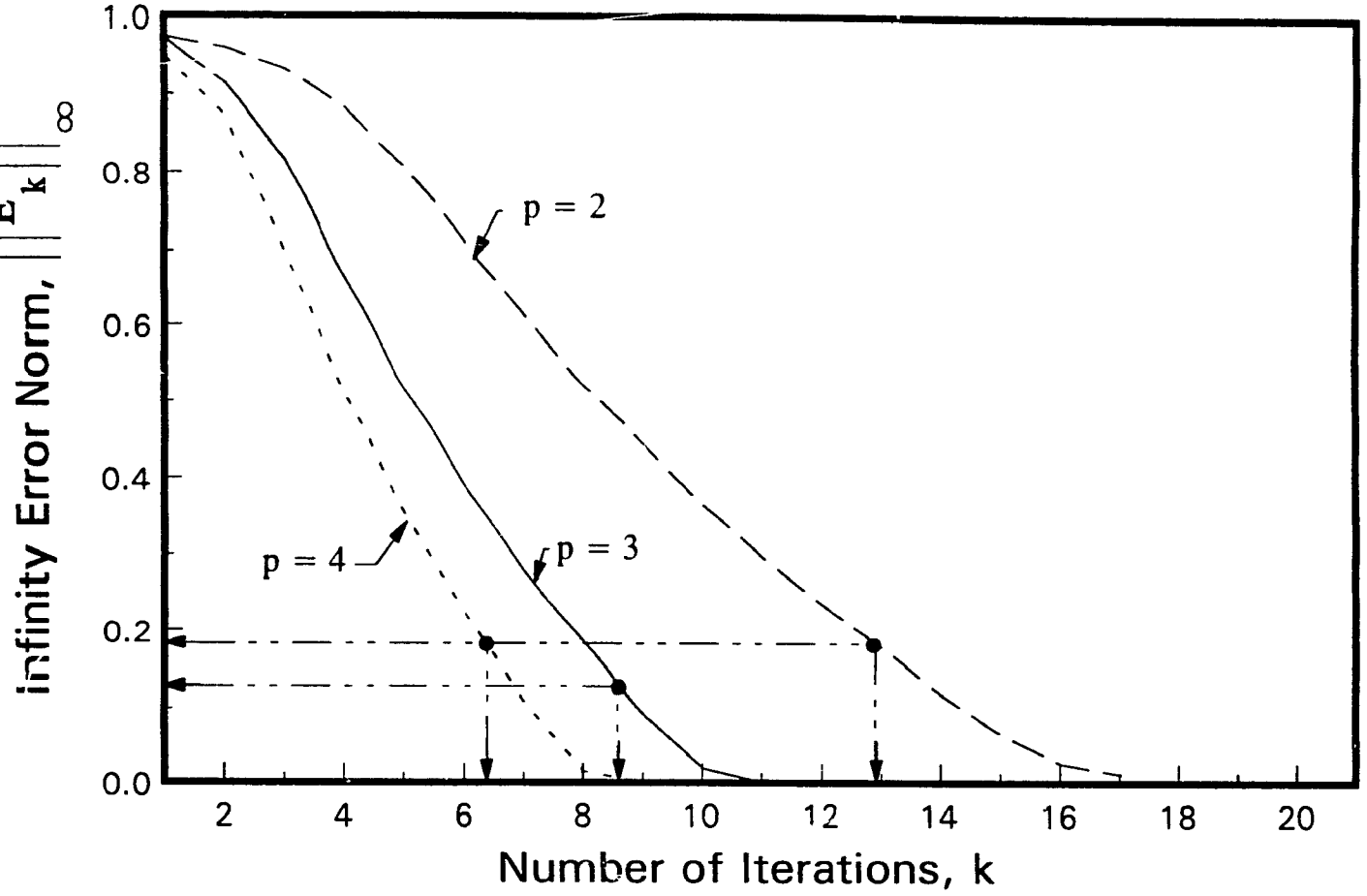

Fig. 3.2 Convergence for various values of p for initial inverse using infinity norm, n = 25x25

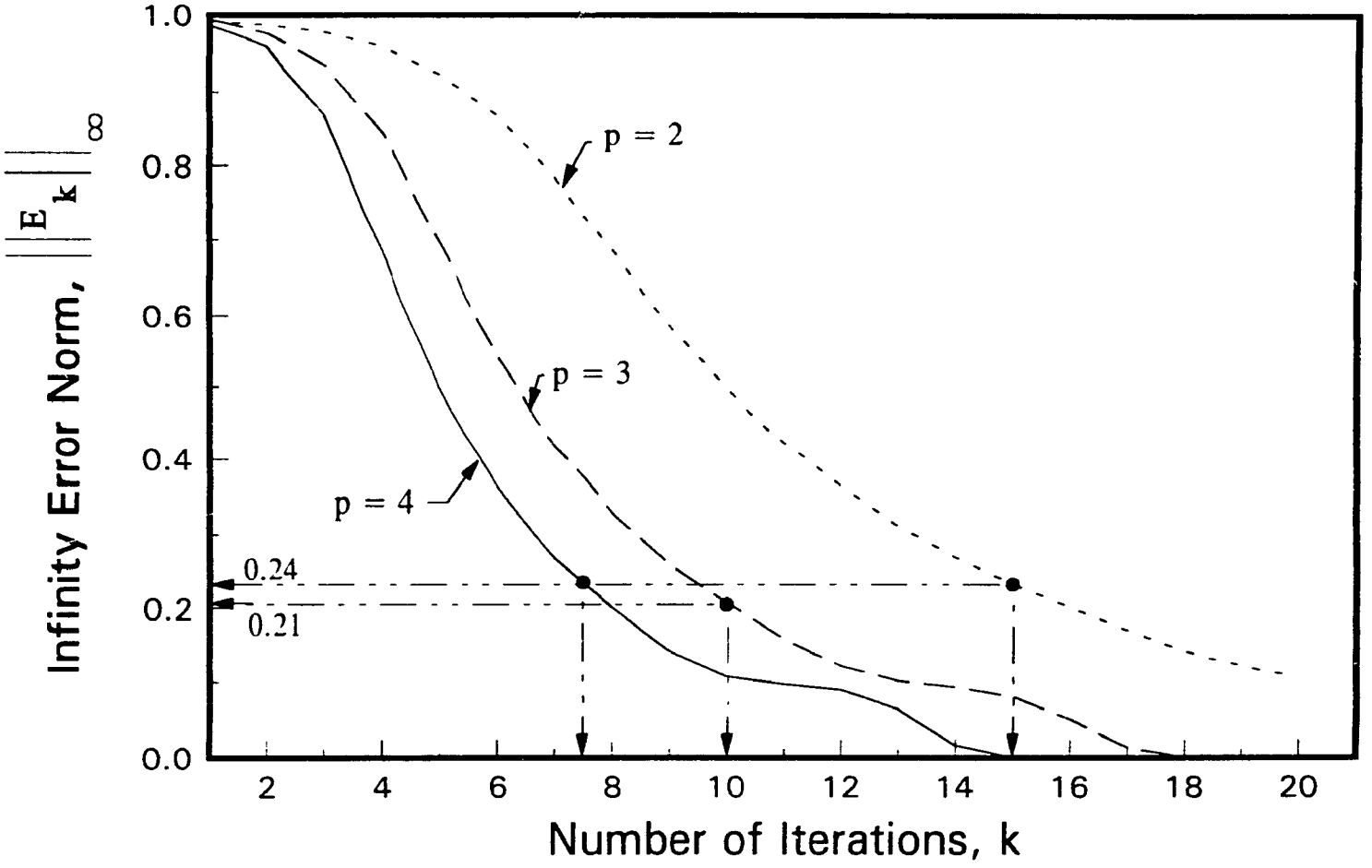

Fig. 3.3 Convergence for various values of p for initial inverse using infinity norm, n = 100x100

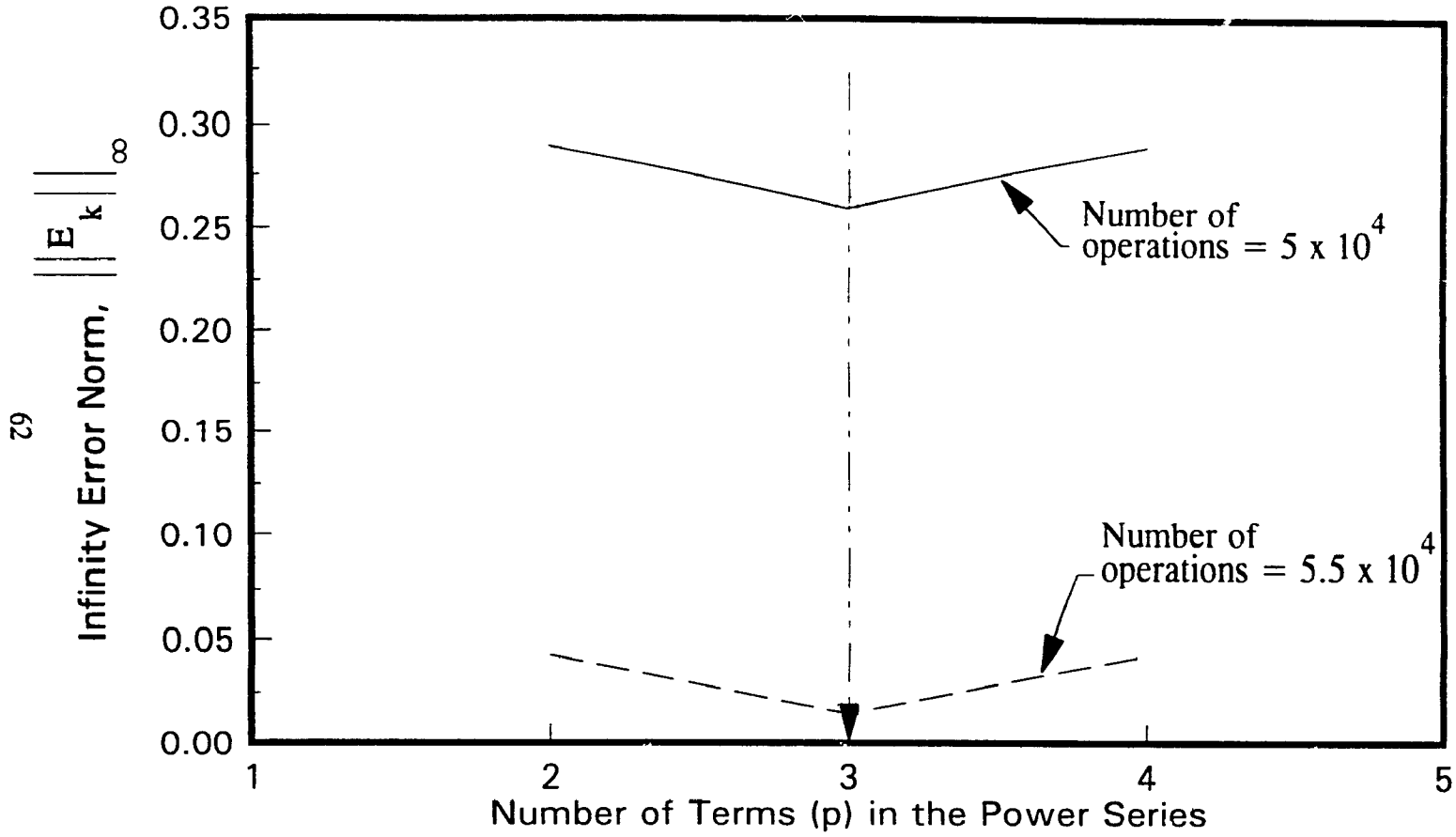

Fig. 3.4a Error Norm Versus Number of Terms, N=10x10

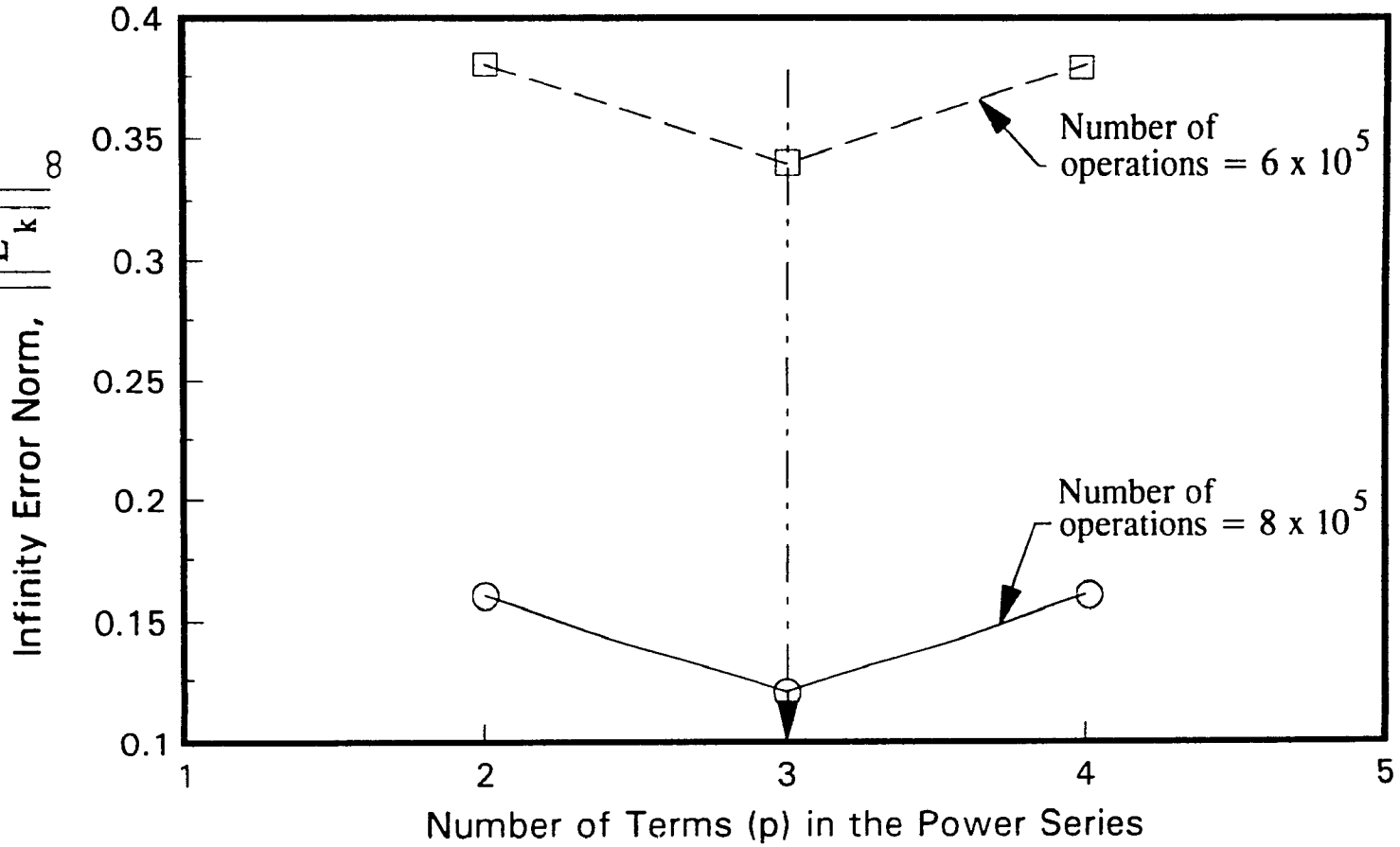

Fig. 3.4b Error Norm Versus Number of Terms, N=25x25

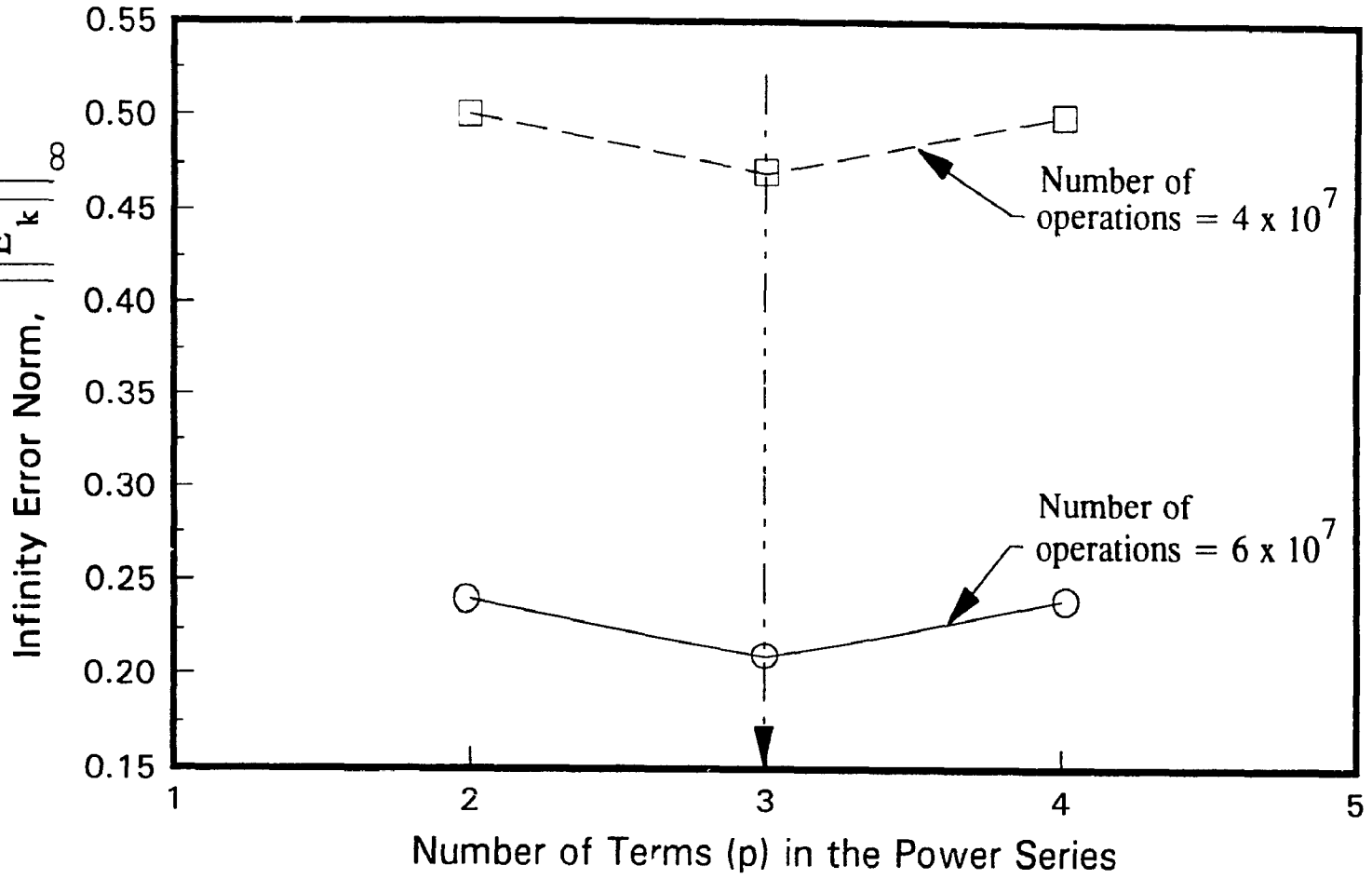

Fig. 3.4c Error Norm Versus Number of Terms, N=100x100

It is observed that the function monotonically decreases for integer values of p from 0 to 3, and increases monotonically thereafter. Hence, p=3 is the only critical point (minimum) in this function. Thus, the experimental results verify that the choice of p = 3, ensures fastest convergence to the desired inverse for a fixed amount of operations.

Total Number of Operations	p = 2		p = 3		p = 4	
	Iteration	$\|E_k\|_\infty$	Iteration	$\|E_k\|_\infty$	Iteration	$\|E_k\|_\infty$
6×10^7	15	0.21	10	0.24	7.5	0.21
4×10^7	10	0.5	6.7	0.47	4.7	0.5

Table 3.3 Number of Iterations Required and Error Norm for Fixed Amount of Operations for Various Values of p Using Infinity Norm, n = 100x100.

To experimentally verify that the initial inverse formulation presented in Chapter 2 is correct, tests are carried out on matrices of order 10x10, 25x25, and 100x100. These matrices are randomly generated and the initial inverse is calculated using the expressions developed in Chapter 2 for various norms (one, infinity and Frobenius). This initial inverse is iteratively improved using the procedure developed in Chapter 2 for different values of p (p = 2, 3 and 4). The results of the tests for a 10x10 matrix are shown in Figs. 3.5a-3.5c for p = 2, 3 and 4, respectively. Figures 3.6a-3.6c and 3.7a to 3.7c show

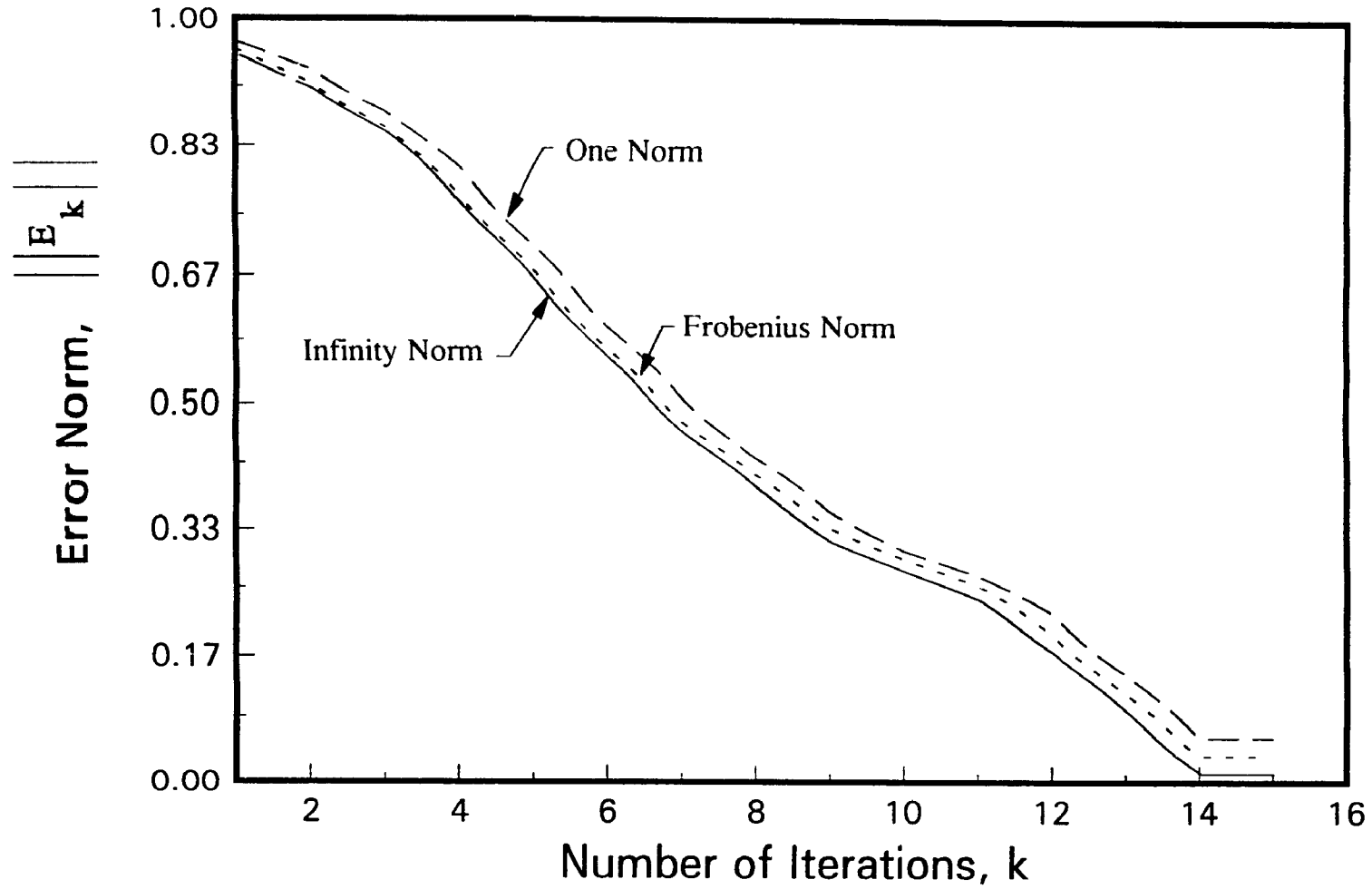

Fig. 3.5a Convergence comparison using different initial inverse formulations, n = 10x10 and p = 2

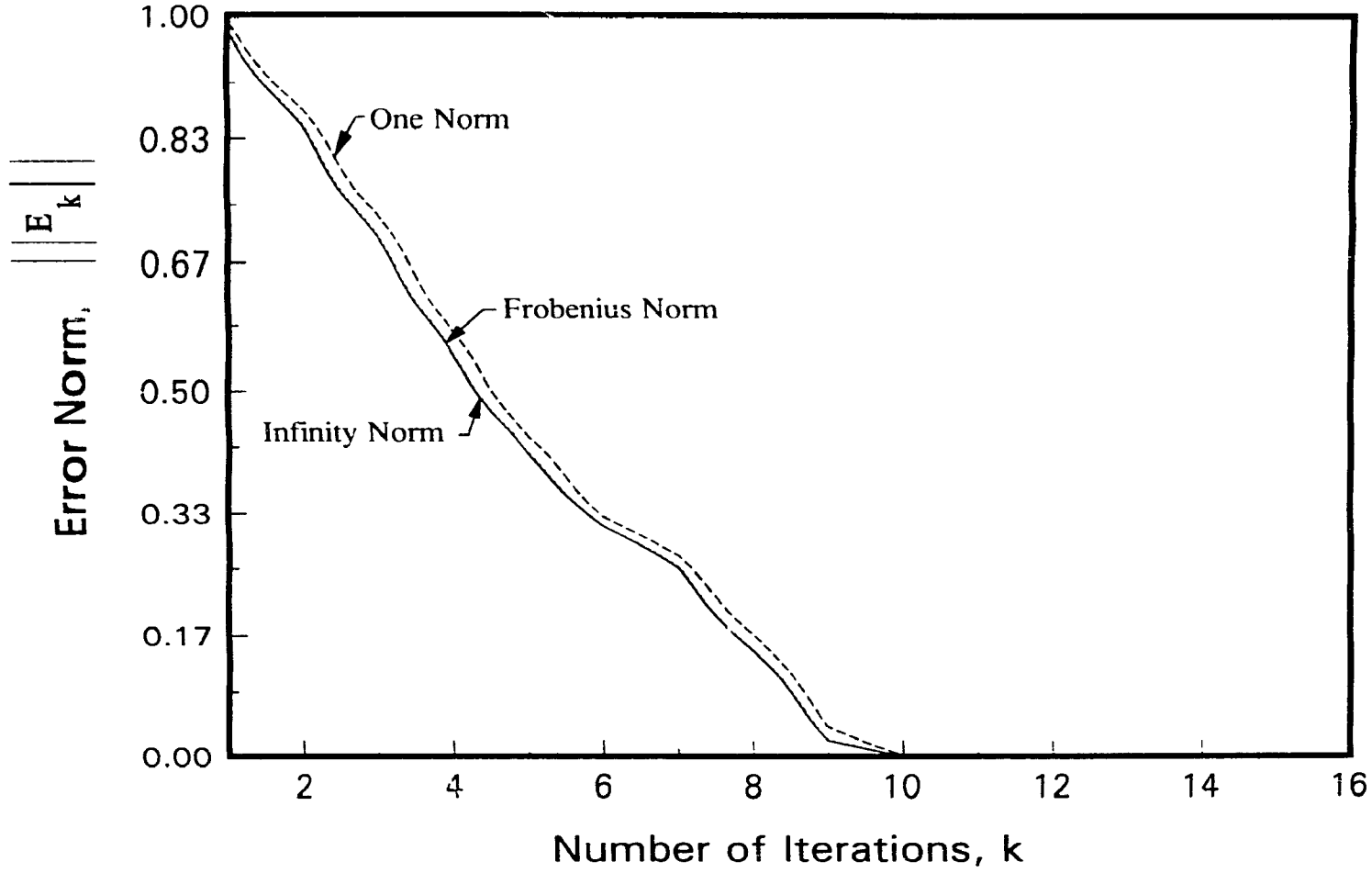

Fig. 3.5b Convergence comparison using different initial inverse formulations, n = 10x10 and p = 3

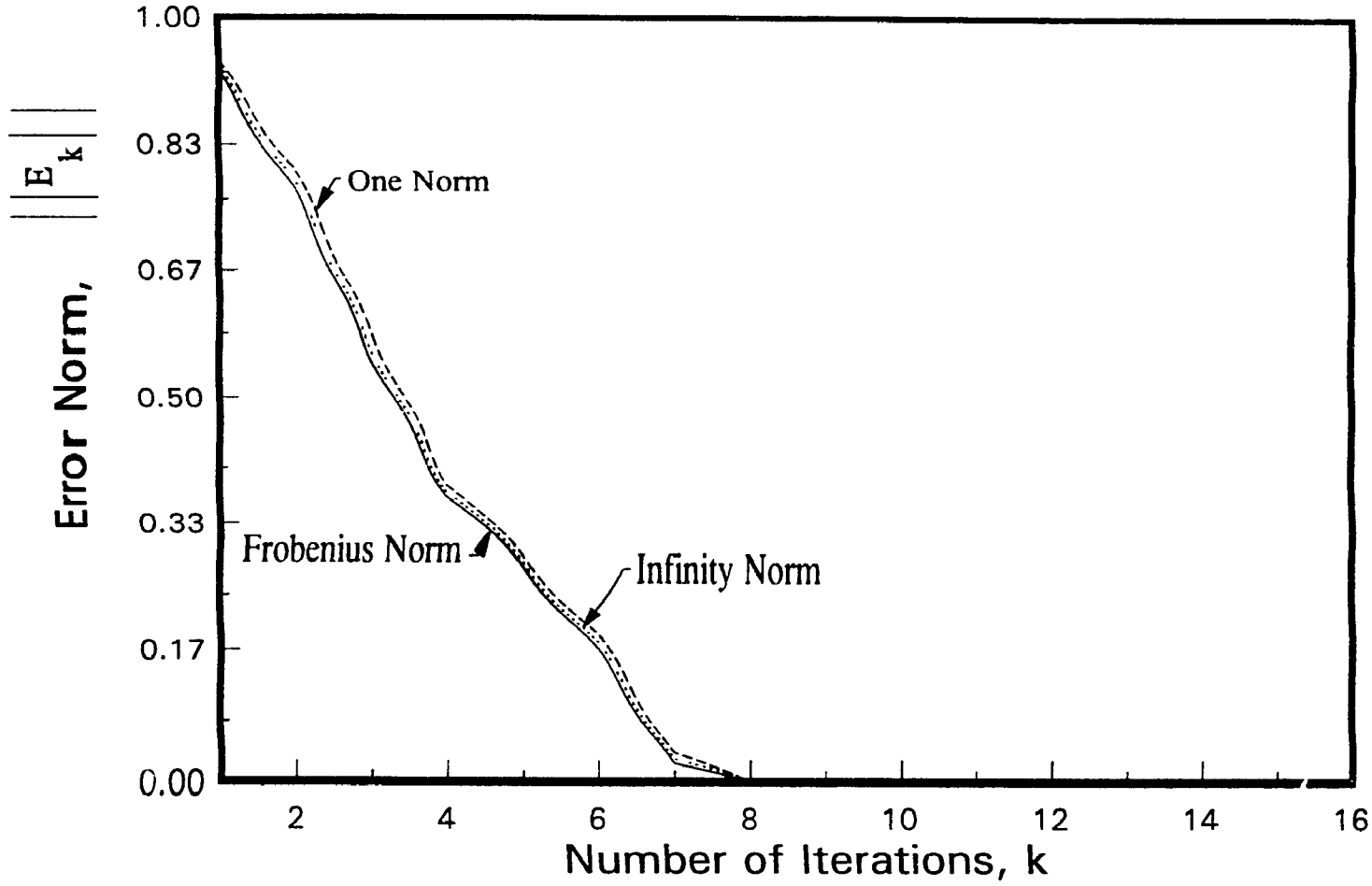

Fig. 3.5c Convergence comparison using different initial inverse formulations, n = 10x10 and p = 4

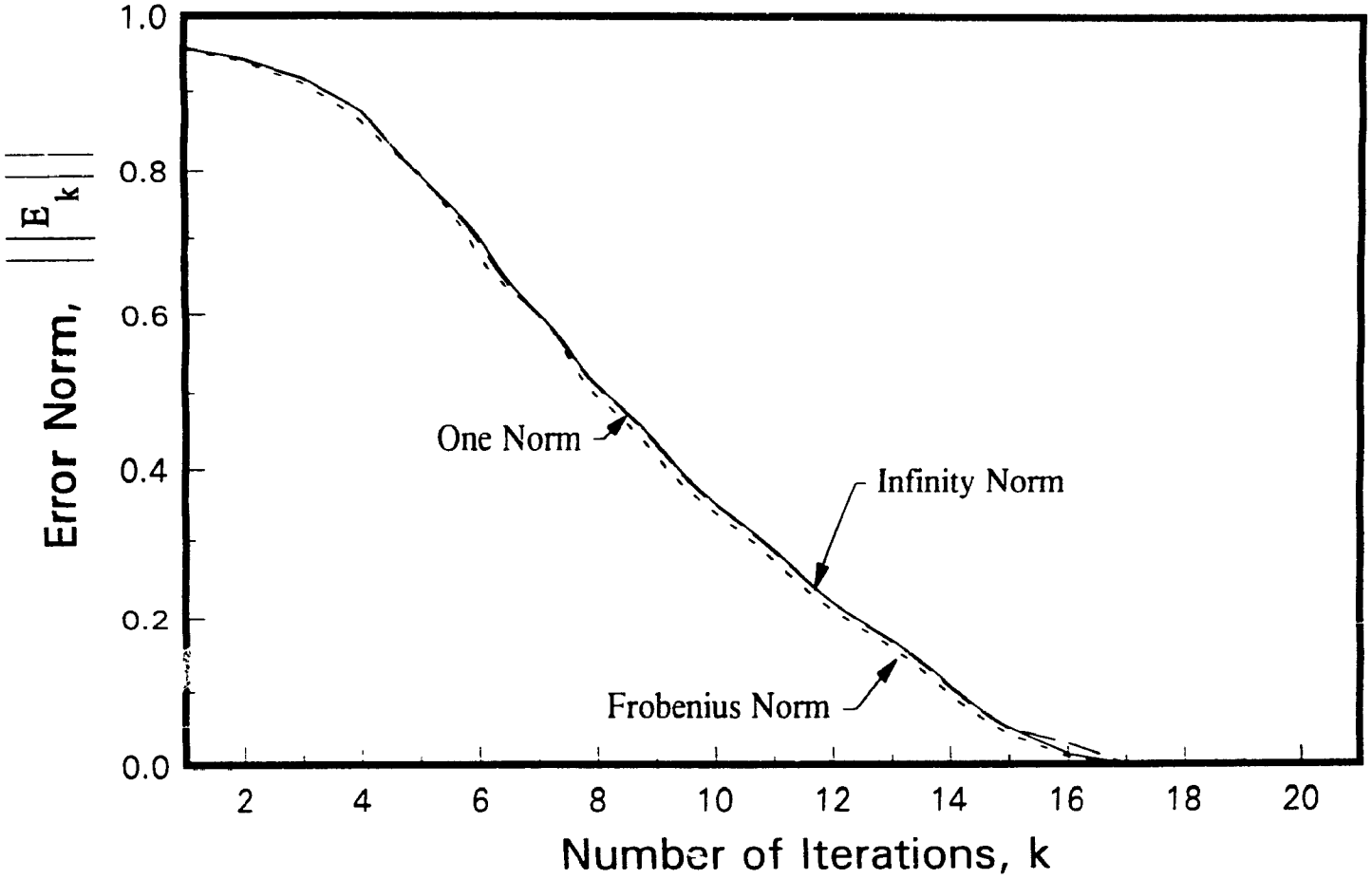

Fig. 3.6a Convergence comparison using different initial inverse formulations, n = 25x25 and p = 2

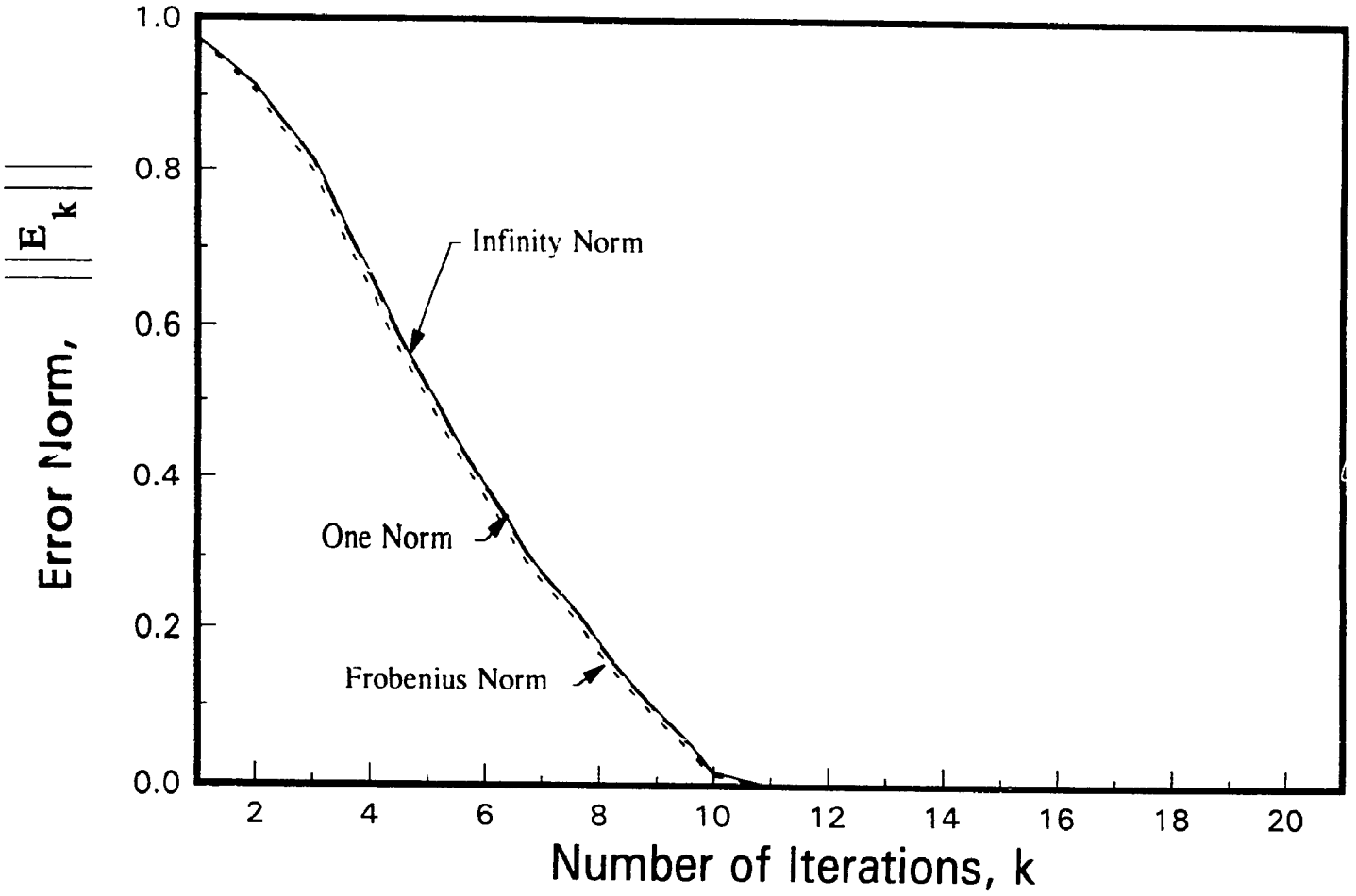

Fig. 3.6b Convergence comparison using different initial inverse formulations, n = 25x25 and p = 3

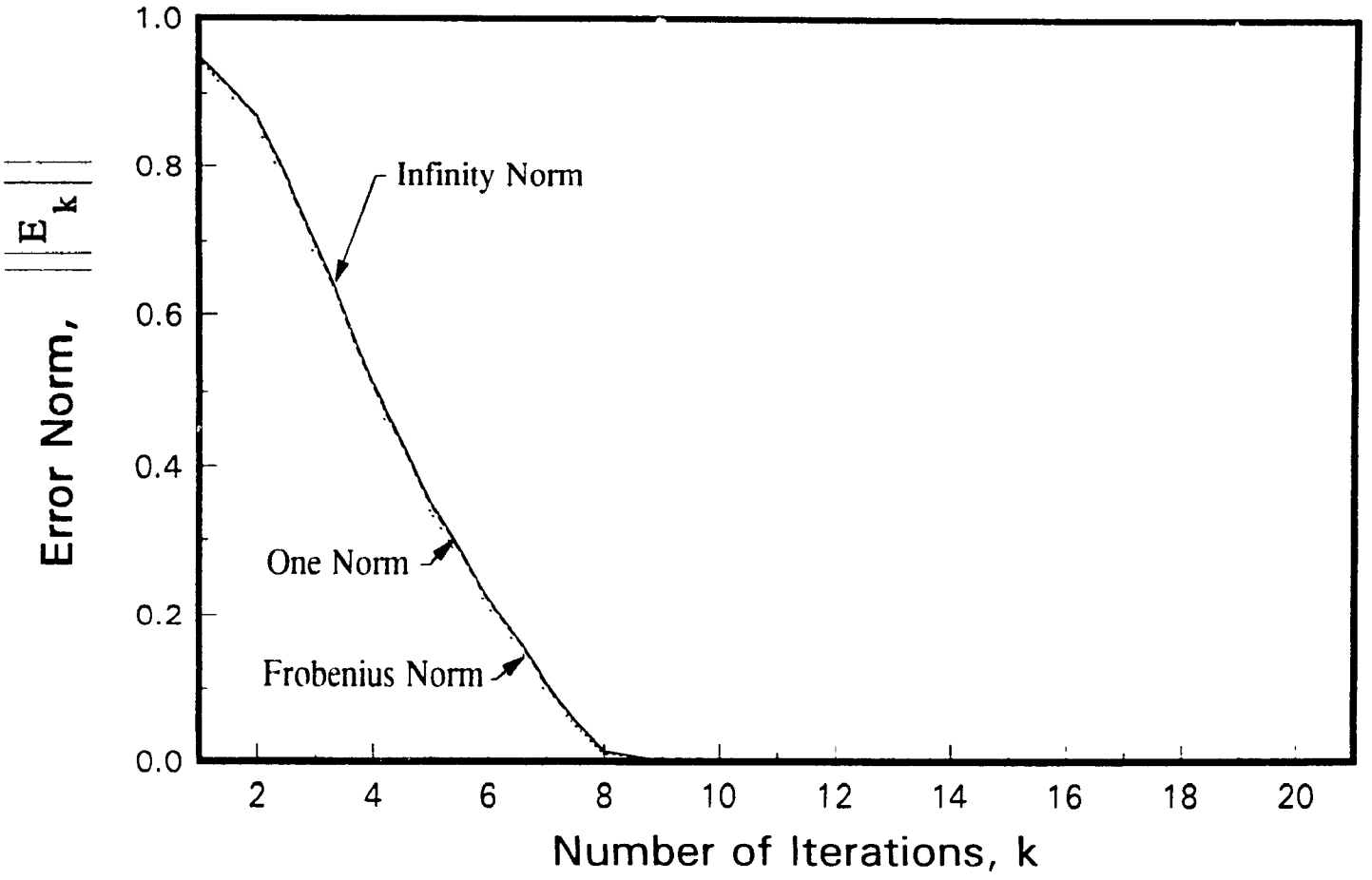

Fig. 3.6c Convergence comparison using different initial inverse formulations, n = 25x25 and p = 4

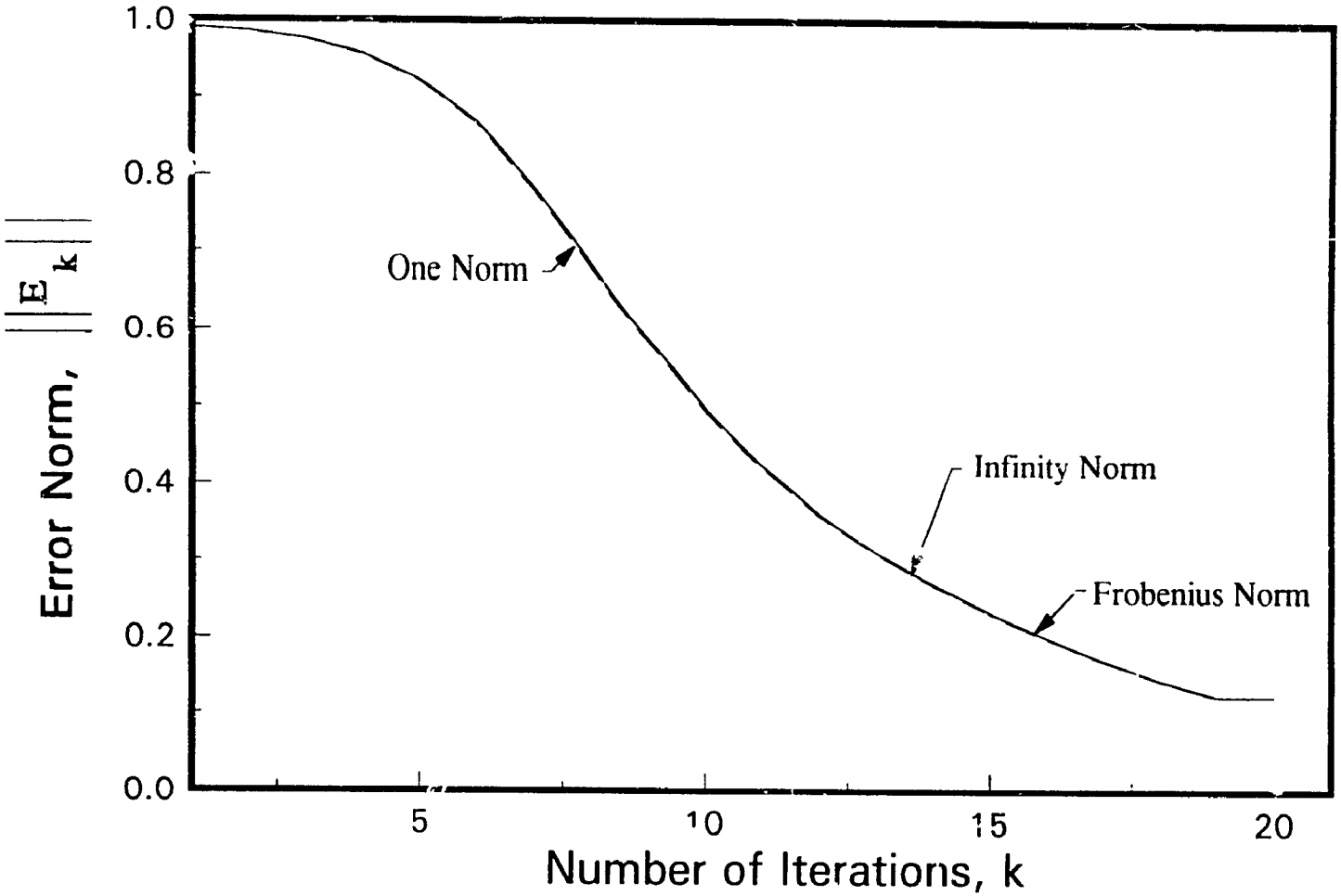

Fig. 3 7a Convergence comparison using different initial inverse formulations, n = 100x100 and p = 2

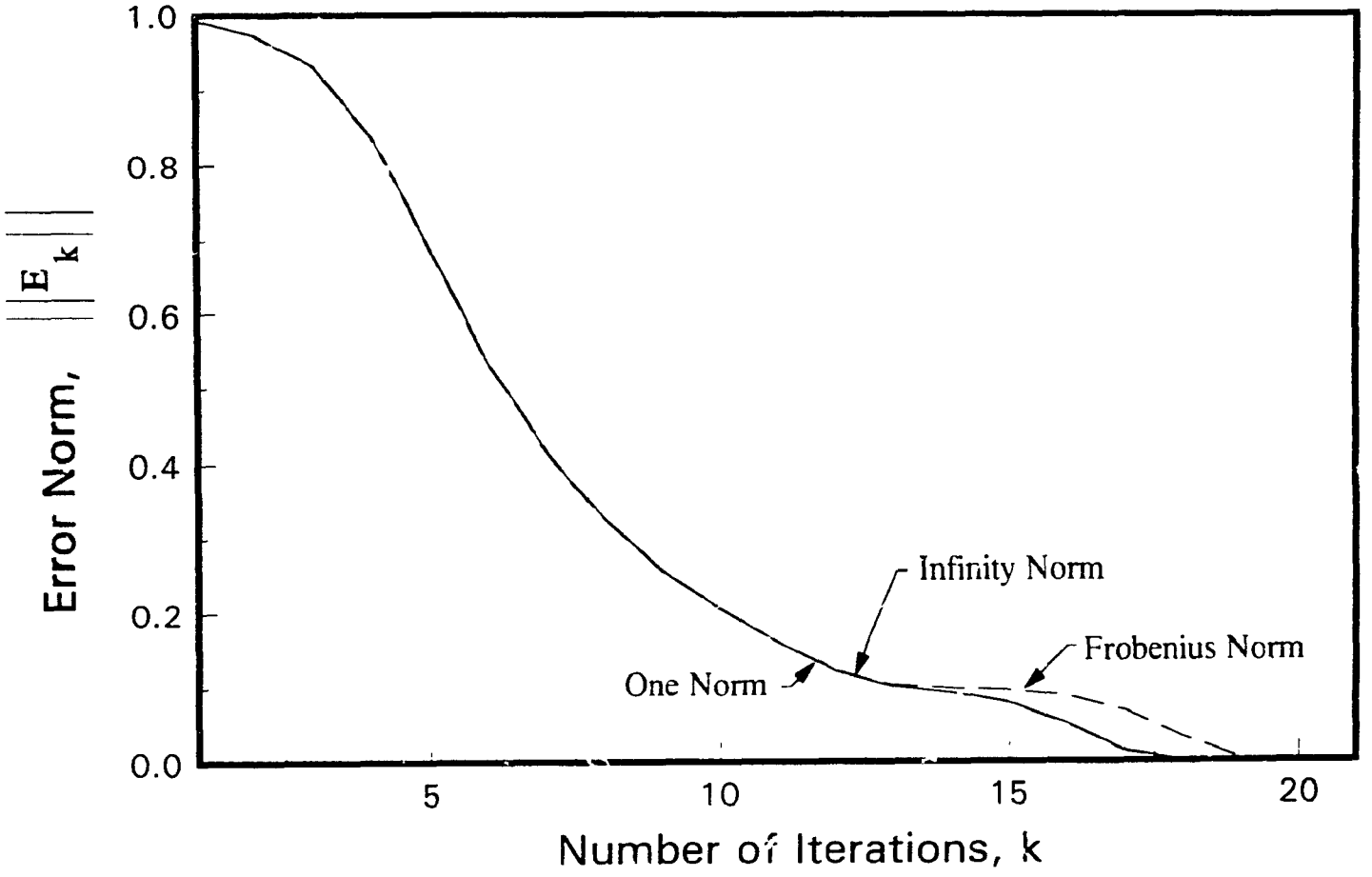

Fig. 3.7b Convergence comparison using different initial inverse formulations, n = 100x100 and p = 3

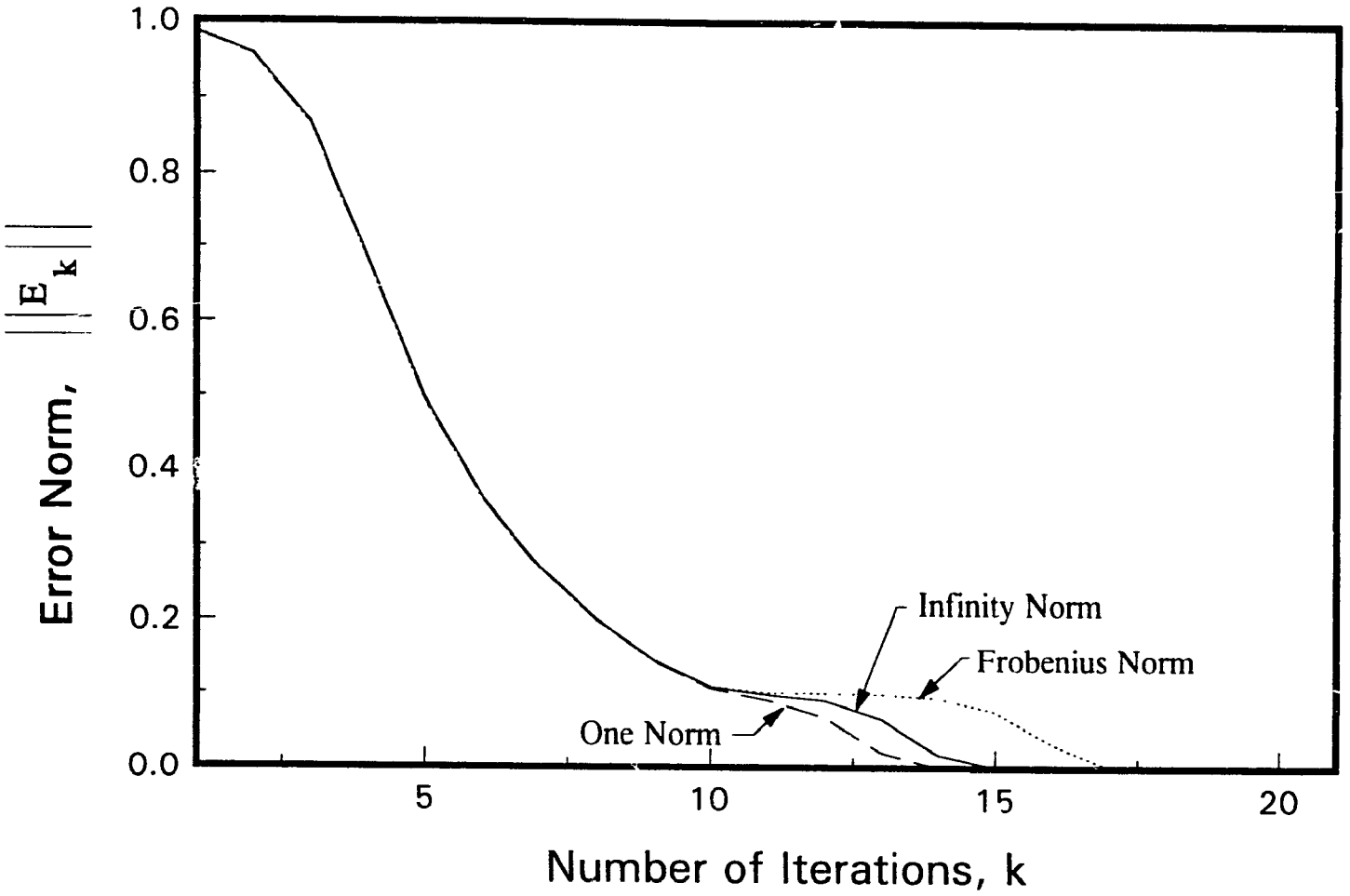

Fig. 3.7c Convergence comparison using different initial inverse formulations, n = 100x100 and p = 4

the results for 25x25 and 100x100 matrices. It can be concluded from Figs. 3.5-3.7, that the method of choosing the initial inverse R_0 is correct since all the results presented exhibit convergence. Further, it is also observed that there is very little difference in the convergence to the actual inverse when different norms are used to formulate the initial inverse and when different orders of matrices are used.

Figure 3.8 shows the effect of the change in the order of the input matrix on the convergence rate for p = 3 using the infinity norm to formulate the initial inverse. It is seen from this figure, that the number of iterations needed for a matrix of order 10x10 such that $\| E_k \|_\infty$ becomes 0 is 9, while it is 11 iterations for a 25x25 matrix. The number of iterations needed for a 100x100 matrix is 17. A ten-fold increase in the order of the matrix does not result in a drastic increase in the number of iterations. Hence, it can be concluded that, the number of iterations is not sensitive to the order of the matrix.

The various tests carried out and the results presented in this section confirm the validity of the theory developed in Chapter 2.

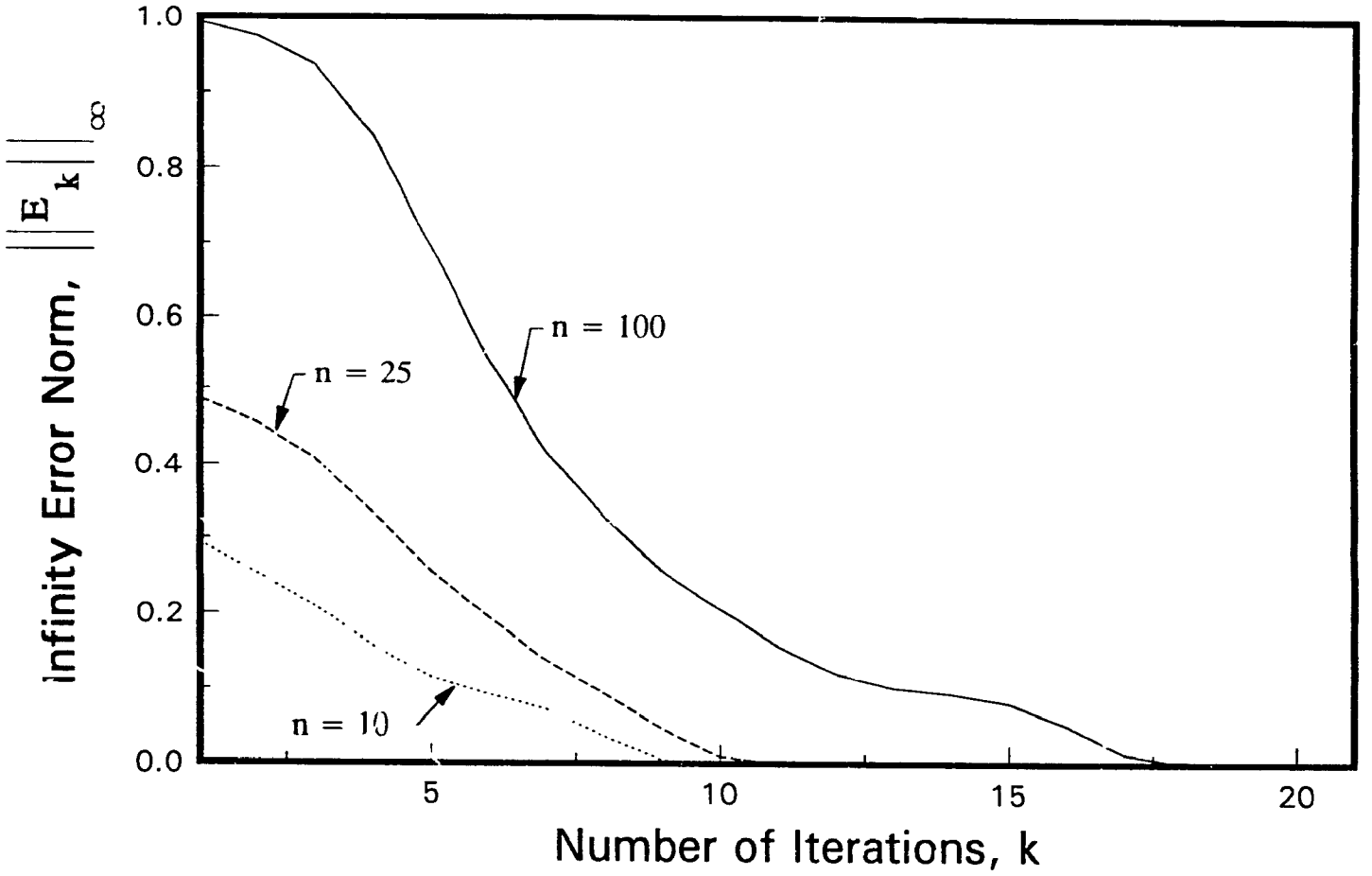

Fig. 3.8 Convergence for Various Matrix Orders with p=3 using Infinity Norm for Initial Inverse

3.6 Summary

This chapter has presented the results of experiments carried out to test the theory developed in Chapter 2. The hardware platform chosen for the implementation is a T-800 transputer, and Occam has been selected as the programming language. The software developed for implementation has been elaborated along with the salient features of the hardware. Tests have been carried out to verify the optimum choice of the number of terms in the power series, and to show that the convergence is assured for the initial inverse chosen based on the schemes developed using the one, infinity and Frobenius norms. Test results have indicated that the number of iterations needed to compute the inverse for a fixed p, is not sensitive to the order of the matrices. It has also been demonstrated that the choice of the initial inverse based on the one, infinity or Frobenius norm has little or no effect on the convergence of the iterative procedure to obtain the desired inverse.

CHAPTER 4

CONCLUSIONS

AND

SUGGESTIONS FOR FUTURE WORK

4.1 Conclusions

The focus of this thesis has been the development of an efficient, generalized iterative algorithm specifically suited for the inversion of large matrices. The efficiency of an algorithm is ensured if the number of operations involved in the computations are minimum. The formulation presented in this thesis makes use of reduced number of operations compared to the general iterative scheme involving an infinite number of terms. The optimum number of terms needed from the infinite series for a given amount of computations has been deduced in order to further improve the efficiency of the proposed iterative scheme. Iterative schemes arrive at the desired solution by continually improving an initial value. Hence, initial inverses have been formulated using different norms.

The generalized, efficient formulation for matrix inversion developed depends only on the initial inverse matrix and the initial residual matrix, $\mathbf{E}_0$. This approach reduces the number of computations involved to calculate the inverse, as the calculations of the residual matrix $\mathbf{E}_k$ and its powers in each iteration are not needed. This scheme has

reduced the operational count by $n^2(k-1)$, where k is the number of iterations and n is the order of the matrix, compared to the conventional approach in which the residual matrix in each iteration is computed. Investigation of the analytical results reveals that the optimum choice for the number of terms from the power series is three for a fixed number of arithmetic operations. From the generalized expression for the iterative scheme for matrix inversion, a formulation for p = 3 has been deduced. Initial inverse formulations have been devised using the one, infinity and Frobenius norms such that the resultant residual matrices are convergent. A stopping criterion has been established to determine the number of iterations needed, given the allowable percentage deviation in the computed inverse. From this criterion, it has been deduced that the following factors affect the stopping criterion : (i) the norm of the initial residual matrix; (ii) the choice of the number of terms (p) and (iii) the number of iterations used.

To verify the concepts developed in Chapter 2, tests have been carried out on a transputer using Occam. Test results illustrate the convergence of the generalized expression developed for the iterative scheme to the actual inverse for various values of the number of terms (p = 2, 3 and 4) in the infinite series. The experimental results demonstrate that the choice of 3 terms from the power series for a fixed amount of computations is optimal. The correctness of the initial inverse formulations is also verified. However, it is observed that the choice of the one, infinity, or Frobenius norm in the determination of the initial inverse makes little difference on the convergence of the algorithm. Simulation results have indicated that the number of iterations required

for convergence is not very sensitive to the matrix order. It has been noted that on an average, a ten-fold increase in the order of the matrix, results only in a two-fold increase in the number of iterations for a given value of p.

4.2 Scope for Future Investigation

The initial inverse formulation presented in the thesis is valid for any invertible matrix. However, properties of special matrices may be exploited to formulate initial inverses, which may result in faster convergence to the desired inverse.

The iterative scheme has been implemented on a single transputer. The components of the algorithms developed in this thesis need to be further investigated for parallel implementation on a transputer array. Parallel implementation should address issues of partitioning the data for parallelism and parallelizing the iterations. The computations have to be carefully decoupled so that parallelism can be exploited while still avoiding data dependencies to the maximum extent so that the interprocessor communication overheads can be minimized.

REFERENCES

1. Gabel, R.A., and Roberts, R.A.: "*Signals and Linear Systems*", John-Wiley and Sons, New York, 1992.

2. Fabian, D., Nocetti, G., and Fleming, P.J., "*Parallel Processing in Digital Control*", Advances in Industrial Control, Springer-Verlag, pp. 6-144, 1992.

3. Steriti, R.J., and Fiddy, M.A., "*Regularized Image Reconstruction Using SVD and a Neural Network Method for Matrix Inversion*", IEEE Transactions on Signal Processing, Vol. 41, No.10, Oct. 1993.

4. Dorf, R.C.: "*The Electrical Engineering Handbook*", CRC Press, Ann Arbor, 1993.

5. Brow, R.G., and Hwang, P.Y.C., "*Introduction to Random Signals and Applied Kalman Filtering*", Second Edition, John Wiley & Sons, New York, 1992.

6. Milovanovic, I.Z., Kovacevic, M.A., Stojcev, M.K. and Milovanovic, E.I., "*A Direct Method for BBD Matrix Inversion*", Cybernetics and Systems Analysis, Vol. 30, No. 2, 1994.

7. Welstead, S.T., "*Real-time Iterative Algorithms for Optical Signal Processing*", Real-time Signal Processing, SPIE, Vol. 827, pp. 137-144, 1987.

8. Wink, D.J., "*The Use of Matrix Inversion in Spreadsheet Programs To Obtain Chemical Equations*", Journal of Chemical Education, Vol. 71, No. 6, pp. 490-492, June 1994.

9. Zomaya, A.Y. : "*Modelling and Simulation of Robot Manipulators - A Parallel Processing Approach*", World Scientific Series in Robotics and Automation Systems - Vol. 8, USA, 1992.

10. Gallivan, K.A., Plemmons, R.J., and Sameh, A.H., "*Parallel Algorithms for Dense Linear Algebra Computations*", Parallel Algorithms for Matrix Computations, pp 1-80, 1990.

11. Milovanovic, E.I., Milovanovic, I.Z. and Stojcev, M.K., "*Matrix Inversion Algorithm for Linear Array Processor*", Parallel Processing: CONPAR 92-VAPP V. Second Joint International Conference on Vector and Parallel Processing, pp. 367-372, 1992.

12. Schlereth, F.H., and Schlereth, B.F., "*Kilonode - A Transputer-based Parallel Computer*," Transputer Research and Applications 1, NATUG-1 Proceed. of the First Conf. of the North American Transputer Users Group, Utah, pp. 82-88, April 1989.

13. Schlereth, F.H., and Schlereth, B.F., "*Parallel Routing Algorithms for Kilonode*", Transputing 91, Proceed. of the World Transputer User Group Conf., pp. 68-77, 1991.

14. Pan, V., and Reif, J., "*Efficient Parallel Solution Of Linear Systems*", Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing, pp. 143-152, May 1985.

15. Isaacson, E., and Keller, H.B., : "*Analysis of Numerical Methods*", John Wiley, 1966

16. Jaja, J. : "*An Introduction to Parallel Algorithms*", Addison-Wesley, 1992.

17. Jain, M.K., Iyengar, S.R.K., and Jain R.K. : "*Numerical Methods for Scientific and Engineering Computation*," Wiley Eastern Limited, pp. 72-102, 1985.

18. El-Amawy, A., and Dharmarajan, K.R., "*Parallel VLSI Algorithm for Stable Inversion of Dense Matrices*", IEEE Proceedings, No. 136, pp. 575-580, 1985.

19. Geus, L., Henning, W., Vajteri, M. and Valkert, J., "*Matrix Inversion Algorithm for a Pyramidal Multiprocessor System*", Computers and Artificial Intelligence, No. 7, pp. 65-79, 1988.

20. Householder, A.S. : "*The Theory of Matrices in Numerical Analysis*", Ginn (Blaisdell), Boston, MA, 1964.

21. Gianey, R.J., "*An Implementation of Parallel Matrix Inverse Algorithm on a Transputer Array*", The Transputers in Australia 2, Proceed. of the 4th Australian Transputer and OCCAM User Group Conf., ATOUG-4, IOS Press, pp. 61-66, 1991.

22. Strassen, V., "*Gaussian Elimination is not Optimal*", Numerische Mathematik, Vol. 13, pp. 354-356, 1969.

23. Pease, M.C., "*Inversion of Matrices by Partitioning*," J of the Association for Computing Machinery, Vol. 16, No. 2, pp. 302-314, April 1969.

24. Levin, M.D., and Evans, D.J., "*The Inversion of Matrices By the Double-bordering Algorithm on MIMD Computers*," Parallel Computing, No. 17, pp. 591-602, 1991.

25. Csanky, L., "*Fast Parallel Matrix Inversion Algorithms*", SIAM J of Computing, Vol. 5, No. 4, pp. 618-623, Dec. 1976.

26. Faddeev, D.K., and Faddeva, V.N. : "*Computational Methods of Linear Algebra*", W.H. Freeman, San Francisco, 1963.

27. Lakshmivarahan, S., and Dhall, S.K., "*Analysis and Design of Parallel Algorithms : Arithmetic and Matrix Problems*", McGraw-Hill Series on Supercomputing and Parallel Processing, 1990.

28. Golub, G.H., and Van Loan, C.F. : "*Matrix Computations*," The John Hopkins University Press, 1990.

29. Pan, V., "*Complexity of Algorithms for Linear Systems of Equations*", Computer Algorithms for Solving Linear Algebraic Equations, pp. 27-52, 1991.

30. Winograd, S., "*A New Algorithm for Inner Product*," IEEE Transactions on Computers, pp. 693-694, July 1968.

31. Pan, V., "*How Can We Speed Up Matrix Multiplication?*", SIAM Review, Vol. 26, No. 3, pp. 393-415, July 1984.

32. Harp, G. : "*Transputer Applications*", Pitman Computer System Series, 1989.

33. Burns, A., "*Programming in Occam 2*", Addison Wesley Publications, 1990.

34. Chaudhuri, P. : "*Parallel Algorithms - Design and Analysis*", Advances in Computer Science Series, Prentice Hall, 1992.

35. Francomano, E., Pecorella, A., and Macaluso, T.A., "*Use of the Matrices Products in the Inverse Matrix Computation*," Conf. on Parallel Computing: Problems, Methods and Applications, Italy, 1992.

Appendix A

Maple Results for Symbolic Computation

This appendix shows the proof that has been obtained using a symbolic manipulation software MAPLE. These results are summarized in equations (2.36) to (2.42). In the following equations the variable k is equal to λ_s in the above mentioned equations

Step - 1 : Function entry $a1 = k^{p^{1/p}}$

```
> a1:=k^(p^(1/p));
```

$$a1 := k^{(p^{(1/p)})}$$

Step - 2 : Find the derivative of the function with respect to p: $a2 = \frac{\delta a1}{\delta p}$

```
> a2:=diff(a1,p);
```

$$a2 := k^{(p^{(1/p)})}\, p^{(1/p)} \left(-\frac{\ln(p)}{p^2} + \frac{1}{p^2} \right) \ln(k)$$

Step - 3 : Simplify the derivative.

```
> a3:=simplify(a2);
```

$$a3 := -\frac{k^{(p^{(1/p)})}\, p^{(1/p)} (\ln(p) - 1) \ln(k)}{p^2}$$

Step - 4 : Solve the first derivative for p $a4 = \frac{\delta a1}{\delta p} = 0$

```
> a4:=solve(a3=0);
                a4 := {k = 1, p = p}, {k = k, p = exp(1)}
```

Solution is : p = e[1]

Step - 5 : Find the second derivative of p : $a5 = \frac{\delta^2 a3}{\delta p^2}$

```
> a5:=diff(a3,p);
```

$$a5 := -\frac{k^{(1/p)}\left(p^{(1/p)}\left(\frac{1}{p^2}+\frac{\ln(p)}{p^2}\right)\ln(k)\,(\ln(p)-1)\right)}{p^2}$$

$$-\frac{k^{(1/p)}\,p^{(1/p)}\left(\frac{1}{p^2}+\frac{\ln(p)}{p^2}\right)(\ln(p)-1)\ln(k)}{p^2}$$

$$-\frac{k^{(1/p)}\,p^{(1/p)}\ln(k)}{p^3}+2\,\frac{k^{(1/p)}\,p^{(1/p)}\,(\ln(p)-1)\ln(k)}{p^3}$$

Step - 6 : Simplify the derivative function (a5)

```
> a6:=simplify(a5);
                                  /   - 1 + p\
                        (1/p)     |- 2 -------|
                       (p     )   \      p   /          2
                  a6 := k       ln(k) (p             ln(k) ln(p)

                    /   - 1 + p\              /   - 1 + p\
                    |- 2 -------|             |- 2 -------|
                    \      p   /             \      p   /
                - 2 p           ln(k) ln(p) + p           ln(k)

                  / - 1 + 2 p\          / - 1 + 2 p\        / - 1 + 2 p\
                  |- ---------|         |- ---------|       |- ---------|
                  \     p   /     2     \     p   /         \     p   /
                + p           ln(p)  - 2 p           ln(p) + p

                    / - 1 + 3 p\        / - 1 + 3 p\
                    |- ---------|       |- ---------|
                    \     p   / 2       \     p   / 2          /   2
                - 3 p           p  + 2 p           p  ln(p)) / p
                                                            /
```

Step - 7 : Solve the second derivative given by a6 for p = e^1

```
> a7:=evalf(subs(p=exp(1),a6));
                                1.444667861          -9
                a7 := .1353352833 k          ln(k) (.1*10   ln(k) - .5314636O6)
```

Solution is : $0.13534\ k^{1.4447}\ \ln(k)\ (0.1x10^{-9}\ \ln(k) - 0.53146)$

Step - 8 : End.

```
> quit;
```