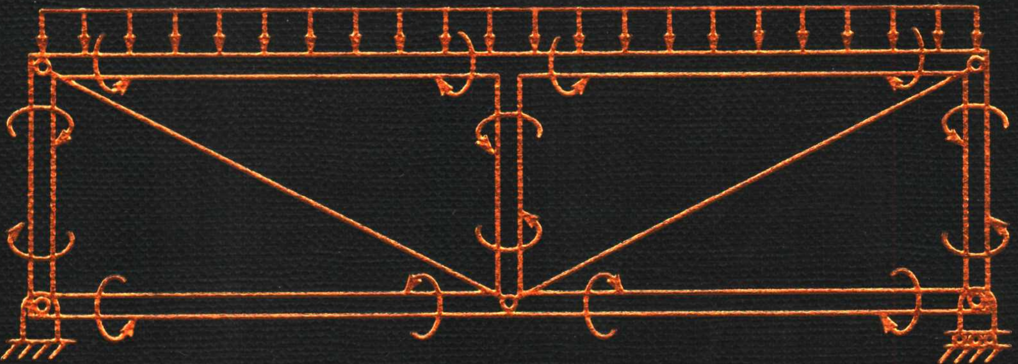
STRUCTURAL ANALYSIS ON MICROCOMPUTERS



CHU-KIA WANG

STRUCTURAL ANALYSIS ON MICROCOMPUTERS

CHU-KIA WANG

Professor of Civil Engineering
University of Wisconsin, Madison

MACMILLAN PUBLISHING COMPANY
New York
COLLIER MACMILLAN PUBLISHERS
London

Macmillan Publishing Company
866 Third Avenue, New York, New York 10022

Collier Macmillan Canada, Inc.

Library of Congress Cataloging in Publication Data

Wang, Chu-Kia,
Structural analysis on microcomputers.

Includes bibliographies and index.
1. Structures, Theory of—Computer programs. 2. Microcomputers—Programming. 3. Basic (Computer program language) I. Title.
TA647.W36 1986 624.1'71'02854 85-5035
ISBN 0-02-424500-3

Printing: 1 2 3 4 5 6 7 8 Year: 6 7 8 9 0 1 2 3 4 5

ISBN 0-02-424500-3

PREFACE

This book provides the basic concepts behind and the explanations for the twelve model computer programs written in the BASIC language. The first two programs, on matrix multiplication and inversion, are included to assist readers in getting acquainted with the particular microcomputer they are using. These two matrix operations are essentially the only ones used in the entire book. Each of the remaining ten programs is self-sufficient to solve problems in structural analysis, from plane trusses to space rigid frames, for limit analysis, and for analysis by subassemblies.

This book is intended to supplement the regular textbooks on structural analysis. Because computer programs can be used to analyze structures of any size and complexity, they can provide, at any level of the course sequence in structural analysis, complete solutions for the problems assigned by the instructor as homework exercises. Furthermore, students can make continual use of these computer programs, or adaptations thereof, for the analysis portion of their design project courses.

At the present time many schools offer one or two courses in structural analysis containing conventional topics as requisites for an advanced course in matrix computer analysis. In such cases the present book can stand alone as the textbook for the course in matrix computer analysis, although the assigned homework should probably be supplemented with exercises that are more variable and complex than those provided in this book. Assignments can also be made to enhance the model computer programs.

As a reference manual for practicing engineers, this text is intended to be a source book for learning the basic concepts of the matrix displacement method and for executing the model computer programs. In most cases the computer programs provided can be applied directly to practical situations. However, once readers can check through the algorithm and the line-by-line statements in any one program, they will be able to modify any program to suit a specific project.

This book is neither an introduction to structural theory nor an advanced text on matrix methods. It is assumed that readers are already knowledgeable in such subjects as moment-area/conjugate-beam theorems, the virtual-work method, and the moment-distribution method. Without invoking the energy theorems, this book provides basic derivations of the matrix displacement method by means of equilibrium, Hooke's law, and compatibility. For those structural engineers who want to write their own computer programs, this book provides a vehicle by which they can learn the basic method through the line-by-line explanations of the model computer programs.

In many respects this book is an enhanced version of the author's *Matrix Methods of Structural Analysis,* first published in 1966 by the International Textbook Company, and in a revised edition in 1970, where FORTRAN programs for the IBM1620 computer were listed in the appendix. The author has found that it is more convenient to write and debug new computer programs in BASIC; if desired, the source program can be compiled in the form of an object program or can be translated into another language (examples of such a translation are provided by the FORTRAN listings in Appendix B of this book) and then compiled.

Experience in teaching this material to groups of practicing engineers has prompted the author to work out the additional examples with computer solutions, provided in Appendix A of this book. In the process of making up input for the problems at hand, most users are aided by the sample input format for these examples. In the event that this book is used as a primary text for a course in matrix computer methods, the instructor may wish to devise new exercises, depending on the level of the course, in the following categories: (1) to require longhand solution of small-degree-of-freedom problems, checking the answers against the computer output; (2) to modify the model computer programs to suit, say, trusses of equal panel length or frames with rectangular joints in large degree-of-freedom problems; or (3) to write enhanced programs either in the interactive mode or in a more user-friendly fashion.

New models of microcomputers that are faster, larger, lighter, and less expensive are coming on the market almost every day. The present book can only be regarded as a first installment, because the author is anxious to share his findings with students and practitioners.

Writers of computer programs know well that typical programs may give correct answers most of the time but may give wrong and unreasonable results when dealing with marginal situations. Inasmuch as the objective of this book is primarily educational, and inasmuch as emphasis has always been placed on checking the output before its use in design, the author does not assume any responsibility for errors produced in the output by the ''nonfoolproof'' programs. Any feedback and suggestions from readers will be greatly appreciated.

C. K. Wang
Madison, Wisconsin

CONTENTS

CHAPTER 6

CHAPTER 7

CHAPTER 8

PROGRAM H Plane Grid Analysis 215

CHAPTER 9

PROGRAM I Space Rigid Frame Analysis 251

CHAPTER 10

CHAPTER 11

CHAPTER 12

APPENDIX A

APPENDIX B

LIST OF EXHIBITS

CHAPTER 1

PROGRAM A

Matrix Multiplication

1.1 Introduction

The two major arithmetical operations in all structural analysis programs described in this book are matrix multiplication and matrix inversion (including solution of simultaneous equations). The short computer program for matrix multiplication, called PROGA (Program A), will rarely be used as such, but it is used many times as parts of other programs. The reader is advised to get this program (PROGA) running smoothly for the purpose of setting up a style for input and output.

1.2 Linear Transformation

To understand what matrix multiplication is, a commonly used term, *linear transformation*, should first be defined.

Consider a set of linear equations in which three values of x are expressed by two values of y, as follows:

$$\begin{aligned} x_1 &= 13y_1 + 5y_2 \\ x_2 &= 7y_1 + 11y_2 \\ x_3 &= 8y_1 + 3y_2 \end{aligned} \tag{1.2.1}$$

Another set of linear equations expresses the two values of y in terms of four values of z, as follows:

$$\begin{aligned} y_1 &= 4z_1 + 9z_2 + 12z_3 + 5z_4 \\ y_2 &= 14z_1 + 6z_2 + 2z_3 + 10z_4 \end{aligned} \tag{1.2.2}$$

In (1.2.1) the y's are the *independent* variables and the x's are the *dependent* variables; in (1.2.2) the y's become the dependent variables and the z's the independent variables.

Based on the information given in (1.2.1) and (1.2.2), it is possible to obtain a third set of linear equations to express the three values of x *directly* in terms of the four values of z. Thus the first two sets are being combined, or transformed, into one set—hence the name "linear transformation." The combined set can be obtained by substituting (1.2.2) in (1.2.1); thus

$$\begin{aligned}
x_1 &= 13y_1 + 5y_1 \\
&= 13(4z_1 + 9z_2 + 12z_3 + 5z_4) + 5(14z_1 + 6z_2 + 2z_3 + 10z_4) \\
&= (13 * 4 + 5 * 14)z_1 + (13 * 9 + 5 * 6)z_2 + (13 * 12 + 5 * 2)z_3 \\
&\quad + (13 * 5 + 5 * 10)z_4 \\
&= (52 + 70)z_1 + (117 + 30)z_2 + (156 + 10)z_3 + (65 + 50)z_4 \\
&= 122z_1 + 147z_2 + 166z_3 + 115z_4 \qquad (1.2.3a)
\end{aligned}$$

$$\begin{aligned}
x_2 &= 7y_1 + 11y_2 \\
&= 7(4z_1 + 9z_2 + 12z_3 + 5z_4) + 11(14z_1 + 6z_2 + 2z_3 + 10z_4) \\
&= (7 * 4 + 11 * 14)z_1 + (7 * 9 + 11 * 6)z_2 + (7 * 12 + 11 * 2)z_3 \\
&\quad + (7 * 5 + 11 * 10)z_4 \\
&= (28 + 154)z_1 + (63 + 66)z_2 + (84 + 22)z_3 + (35 + 110)z_4 \\
&= 182z_1 + 129z_2 + 106z_3 + 145z_4 \qquad (1.2.3b)
\end{aligned}$$

$$\begin{aligned}
x_3 &= 8y_1 + 3y_2 \\
&= 8(4z_1 + 9z_2 + 12z_3 + 5z_4) + 3(14z_1 + 6z_2 + 2z_3 + 10z_4) \\
&= (8 * 4 + 3 * 14)z_1 + (8 * 9 + 3 * 6)z_2 + (8 * 12 + 3 * 2)z_3 \\
&\quad + (8 * 5 + 3 * 10)z_4 \\
&= (32 + 42)z_1 + (72 + 18)z_2 + (96 + 6)z_3 + (40 + 30)z_4 \\
&= 74z_1 + 90z_2 + 102z_3 + 70z_4 \qquad (1.2.3c)
\end{aligned}$$

If it is known that $z_1 = 5$, $z_2 = 2$, $z_3 = 4$, and $z_4 = 3$, the x values can be obtained directly by substituting the z values in (1.2.3); thus

$$\begin{aligned}
x_1 &= 122z_1 + 147z_2 + 166z_3 + 115z_4 \\
&= 122(5) + 147(2) + 164(4) + 115(3) \\
&= 1913 \\
x_2 &= 182z_1 + 129z_2 + 106z_3 + 145z_4 \\
&= 182(5) + 129(2) + 106(4) + 145(3) \\
&= 2027 \\
x_3 &= 74z_1 + 90z_2 + 102z_3 + 70z_4 \\
&= 74(5) + 90(2) + 102(4) + 70(3) \\
&= 1168
\end{aligned}$$

Or, indirectly, the y values can be obtained first by using (1.2.2) and then the x values obtained by using (1.2.1). Thus

$$
\begin{aligned}
y_1 &= 4z_1 + 9z_2 + 12z_3 + 5z_4 \\
&= 4(5) + 9(2) + 12(4) + 5(3) \\
&= 101 \\
y_2 &= 14z_1 + 6z_2 + 2z_3 + 10z_4 \\
&= 14(5) + 6(2) + 2(4) + 10(3) \\
&= 120 \\
x_1 &= 13y_1 + 5y_2 = 13(101) + 5(120) \\
&= 1913 \\
x_2 &= 7y_1 + 11y_2 = 7(101) + 11(120) \\
&= 2027 \\
x_3 &= 8y_1 + 3y_2 = 8(101) + 3(120) \\
&= 1168
\end{aligned}
$$

The fact that the same answers for $x_1 = 1913$, $x_2 = 2027$, and $x_3 = 1168$ are obtained from direct use of (1.2.3) and then from combined use of (1.2.2) and (1.2.1) indicates that (1.2.3) is probably correct. Although one specific application of (1.2.3) for one particular set of z values may not prove conclusively that all coefficients in that equation are correct, one can be reasonably sure when all assumed values of z are nonzero and unequal to each other.

Before trying to debug any computer program, it is of paramount importance that the programmer solve a simple sample problem by longhand and know that the answers are correct.

1.3 Matrix Notation

Equations (1.2.1), (1.2.2), and (1.2.3) can be written in matrix notation as

$$\{x\}_{3\times 1} = [A]_{3\times 2}\{y\}_{2\times 1} \tag{1.3.1}$$

$$\{y\}_{2\times 1} = [B]_{2\times 4}\{z\}_{4\times 1} \tag{1.3.2}$$

$$\{x\}_{3\times 1} = [C]_{3\times 4}\{z\}_{4\times 1} \tag{1.3.3}$$

in which

$$\{x\}_{3\times 1} = \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} \qquad \{y\}_{2\times 1} = \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix} \qquad \{z\}_{4\times 1} = \begin{Bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{Bmatrix}$$

$$[A]_{3\times 2} = \begin{bmatrix} 13 & 5 \\ 7 & 11 \\ 8 & 3 \end{bmatrix}$$

$$[B]_{2\times 4} = \begin{bmatrix} 4 & 9 & 12 & 5 \\ 14 & 6 & 2 & 10 \end{bmatrix}$$

$$[C]_{3\times 4} = \begin{bmatrix} 122 & 147 & 166 & 115 \\ 182 & 129 & 106 & 145 \\ 74 & 90 & 102 & 70 \end{bmatrix}$$

A *matrix* can be defined as a rectangular block of numbers; it is a *column matrix* if it has only one column. A column matrix symbol is enclosed in braces and a rectangular matrix symbol (the word *rectangular* is often omitted), in brackets. The subscripts outside the braces or brackets indicate the number of rows and the number of columns in that matrix. Any particular element in a matrix is represented by the row number and the column number, in that order. Thus for the preceding $[C]$ matrix, $C(2, 3) = 106$ and $C(3, 1) = 74$.

Hereafter in this book the matrices $[A]$, $[B]$, and $[C]$ defined previously will be shown as, for example,

$[C]_{3\times 4} =$

x \ z	1	2	3	4
1	122	147	166	115
2	182	129	106	145
3	74	90	102	70

(1.3.4)

By adding the upper and left labels, plus the horizontal and vertical rules, the fact that (1.3.4) shows the contents of the matrix equation $\{x\} = [C]\{z\}$ becomes clear.

1.4 Matrix Multiplication

Substituting (1.3.2) in (1.3.1) gives

$$\{x\}_{3\times 1} = [A]_{3\times 2}\{y\}_{2\times 1} = [A]_{3\times 2}[B]_{2\times 4}\{z\}_{4\times 1} \qquad (1.4.1)$$

Comparing (1.4.1) with (1.3.3),

$$[C]_{3\times 4} = [A]_{3\times 2}[B]_{2\times 4}$$

or, in general,

$$[C]_{L\times N} = [A]_{L\times M}[B]_{M\times N} \qquad (1.4.2)$$

Thus the matrix $[C]$ is the product of the *premultiplier matrix* $[A]$ and the *post-multiplier matrix* $[B]$. The arithmetic operation of obtaining the product matrix $[C]$ from the multiplier matrices $[A]$ and $[B]$ is called *matrix multiplication*. For matrix multiplication to be possible, the number of columns in the premultiplier matrix must be equal to the number of rows in the postmultiplier matrix—hence the importance of the prefixes *pre* and *post*.

1.5 The Algorithm

In longhand computation, a matrix multiplication can most conveniently be performed in the arrangement shown in Fig. 1.5.1. For instance, the element $C(2, 3)$ can be obtained by drawing a horizontal line through the second row of $[A]$ to intersect the vertical line through the third column of $[B]$. As both lines proceed "inward" for the intersection at $C(2, 3)$, adding the products of the pairs of numbers in the second row of $[A]$ and the third column of $[B]$ gives

$$C(2, 3) = 7 * 12 + 11 * 2 = 84 + 22 = 106 \qquad (1.5.1)$$

The correctness of this procedure is obvious when (1.5.1) is compared with the method by which the coefficient of z_3 in (1.2.3b) is obtained.

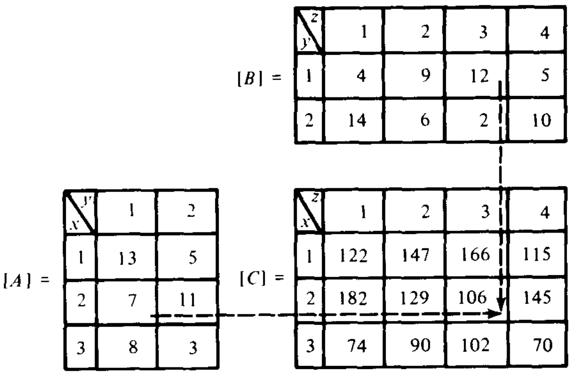


FIGURE 1.5.1 Inner product rule for matrix multiplication.

In computer programming, the word *algorithm* is used for the symbolic expression in general terms of a repetitive process. Thus the algorithm for (1.5.1), in general terms, is

$$C(i, j) = \Sigma A(i, k) * B(k, j) \qquad \text{for } k = 1 \text{ to } M \qquad (1.5.2)$$

Because as k increases from 1 to M the horizontal movement through the ith row of $[A]$ and the vertical movement through the jth column of $[B]$ are both inward, (1.5.2) has been called the *inner product rule* for matrix multiplication.

1.6 The Computer Program

To obtain the product matrix $[C]_{L\times N}$ from the premultiplier matrix $[A]_{L\times M}$ and the postmultiplier matrix $[B]_{M\times N}$, the algorithm has been shown by (1.5.2) to be