MITNE-187

INVESTIGATION OF SOLUTION TECHNIQUES FOR LARGE SPARSE BAND WIDTH MATRIX EQUATIONS OF LINEAR SYSTEMS

by

L.J. Guillebaud

M.W. Golay

May 1976

Massachusetts Institute of Technology

Department of Nuclear Engineering

Cambridge, Massachusetts

INVESTIGATION OF SOLUTION TECHNIQUES FOR LARGE SPARSE BAND WIDTH MATRIX EQUATIONS OF LINEAR SYSTEMS

by

LOUIS J. M. GUILLEBAUD

May 1976

for

Special Problem in Nuclear Engineering

Supervisor: Professor Michael W. Golay

ABSTRACT

Iterative procedures and their computer applications are considered for the problem of calculating intro-bundle cross-flows in PWR cores. The sparse band striped cross-flow coefficients matrix is carefully analyzed for a minimum storage in the core memory. The generated matrix is used in iterative algorithms for solution. Different convergency criteria are discussed. Other possible techniques are presented. A computer routine, based on the iterative procedure developed, and to be included in a large thermal hydraulic analysis code, is detailed. Comparison between the effectiveness of an iterative procedure and the Gauss Elimination Method, in their ability to solve the cross-flow problem, is discussed on the basis of their computer applications.

Supervisor: Michael W. Golay
Title: Associate Professor of Nuclear Engineering

ACKNOWLEDGMENTS

The author would like to express his sincere thanks to Professor M. W. Golay for his invaluable advice and guidance throughout this work.

Special thanks are given to Mr. R. W. Bowring whose experience with the COBRA code has been very helpful for the development of the new methods.

The author also wishes to thank Professor K. F. Hansen and Professor L. Wolf for their helpful comments.

Special thanks are also extended to Sylvain et Sylvette whose advice for computer application was very helpful.

I am deeply grateful to my wife for her excellent typing.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

1.1 Introduction to the Problem

There is a great computational advantage in improving the numerical computational method of matrix inversion, or of the similar problem of solving a set of inhomogeneous linear equations, i.e. solving the vector equation:

$$A \underline{X} = \underline{Y} \qquad (1)$$

for $\underline{X}$, given $\underline{Y}$.

The objective of this work is to develop a method which will improve the cross-flows calculation in a large nuclear reactor thermal hydraulic analysis computer code COBRA [1, 2, 3, 4] by means of an iterative procedure.

If the Gauss Elimination [5] method is considered the costs of computer calculation time for large order matrices can be a serious concern: the number of required arithmetic operations is approximately $(n^3/3 + n^2)$, where n is the order of the matrix, or the number of equations to be solved. Since the total required computer time is directly proportional to the number of operations performed, this can lead to long residence time and then to costly computations. However, note that Gauss Elimination, being a direct method, will give an exact solution if there are no round-off errors.

Now, if as an alternative the Gauss-Siedel iterative procedure[5] is considered, and assuming all the necessary conditions for convergence are satisfied, the calculation time can be reduced significantly. This

method requires $(2n^2 + n)$ operations for one iteration. Of course, several iterations are normally required for an accurate result. Table 1.1 shows the number of Gauss-Siedel iterations which may occur for several matrix sizes when the total number of operations remains less than that in the Gauss Elimination solution. It clearly shows that for small matrices, the Gauss Elimination method would be used since the number of allowed iterations is too low to give accurate results. However, an iterative procedure will be by far superior to the Gauss Elimination method when large matrices are considered: this is the most interesting result since, as is explained later, large matrices will be considered in this work.

Because of these considerations, it has been decided to use an iterative solution technique for the resolution of a linear system of equations relating the COBRA IIIC code cross-flows to the differential pressure in adjacent reactor subchannels through a matrix A formed by the cross-flow coefficients. This problem was originally solved directly by the Gauss Elimination method.

It should be also noted that the motivation of this work arises from the fact that A is not a "classical" full matrix, but as is explained later it is found to be band-striped, symmetric and sparse.

Therefore, it has been thought that an iterative procedure could take into account the particular characteristics of this matrix, resulting in more efficient solutions.

The objectives of this work are the following:

1) development and implementation of an iterative technique for solving the cross-flows problem, which is a part of a large computer code.
2) reduction of the storage requirement for the matrix A by generating it in a compact form in the core memory.

Table 1.1: Comparison between the Number of Operations Required by the Gauss Elimination Method and the Gauss-Siedel Method (for one iteration) for Different Matrix Sizes

n = size of matrix / Solution Technique	12	16	32	128	356
GAUSS ELIMINATION (DIRECT METHOD)	≃ 720	≃ 1621	≃ 11946	≃ 715534	≃ 15166074
GAUSS-SIEDEL METHOD (ITERATIVE PROCEDURE)	≃ 300	≃ 528	≃ 2080	≃ 32896	≃ 253828
nber of "allowed" iterations	2	3	5	21	59

1.2 Background

It is useful to explain the origin of the cross-flow problem. In the thermal hydraulic analysis of a reactor, fluid flow processes and also heat transfer are analyzed by means of the subchannel method. This consists of dividing the rod bundle into individual flow channels which are coupled to their neighbors by cross-flows across adjacent boundaries.

The geometry of the model used to describe a rod bundle in this problem is the following: (See Fig. 1.1)

1. A certain number of axial planes - or axial steps - equally spaced represent the fuel rod bundle from the bottom to the top of the core.
2. Each axial plane representing a certain axial segment of the rod bundle, is subdivided into the same number of identical subchannels. The geometry of the grid is chosen to be that of a square array, i.e. each subchannel in a square of identical dimension. Note that this configuration allows a great flexibility in the treatment of the problem: symmetry consideration, reproduction of the array, and extension to large cases are possible. (See Fig. 1.1)

The problem is mathematically formulated in such a way that in each subchannel it is possible to compute enthalpy, mass-flow, pressure, velocities, mass-fluxes, etc., at each axial station.

In particular the cross-flows are related to differential pressure between two adjacent flow channels by the matrix A, which is formed of the cross-flow coefficients. At each axial step J, the linear system to be solved is then:

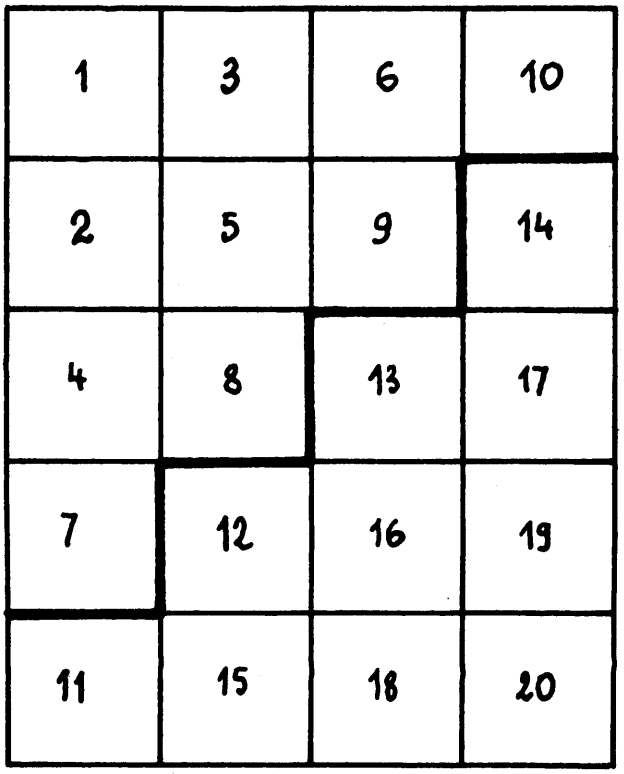


FIG. 1.1: Symmetry Consideration for a 20 Subchannel Case

$$A_J \underline{X}_J = \underline{Y}_J \quad , \quad \text{where}$$

$\underline{Y}_J$ is the vector of differential pressure between two successive axial steps,

$\underline{X}_J$ the cross-flow vector, at axial step J, whose components are the cross-flows at each boundary, and

A_J the cross-flow coefficients matrix at axial step J.

CHAPTER 2
ARRANGEMENT AND DEFINITION OF CROSS-FLOW COEFFICIENTS MATRIX

2.1 Subchannel Considerations

When considering a set of 2 adjacent subchannels I and J, as in Fig. 2.1, the cross-flow at boundary I-J is affected by the six other cross-flows across the remaining boundaries of subchannels I and J. Moreover, it is assumed that cross-flows across all other boundaries have no effect on the particular cross-flow I-J, and they are ignored in the transverse mass and momentum balance which determines the I-J cross-flow.

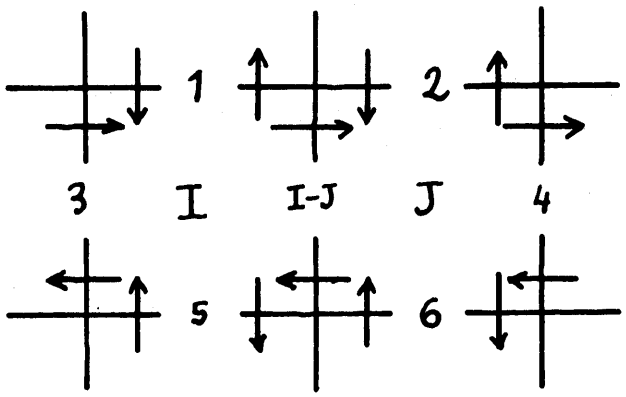
For example, if in this case the total number of boundaries or cross-flows is n, (n-7) cross-flows are set to 0, when considering their effect on I-J.

Recalling now the relation between cross-flows and differential pressure one can write for axial step J:

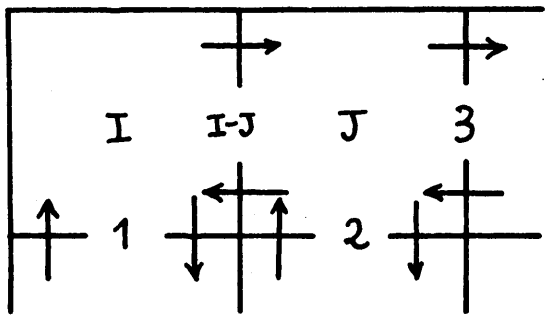
$$\sum_{k=1}^{n} a_{k,J}\, x_{k,J} = \underline{y}_{i,J} \qquad (i=1,\ldots,n) \tag{2.1}$$

where the $x_{k,J}$ are the components of the n-column cross flow vector $\underline{X}_J$.

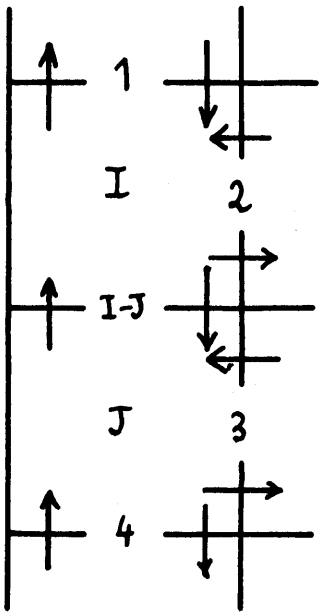
For the previous example, the computation of the sum of the products in the Eq. 2.1 involves only seven elements, since (n-7) products are zero. Then in the array of coefficients $a_{i,k}$ $(1 \leq i, k \leq n)$ of the particular row i, will have seven significant elements. The chosen case corresponds to the most general location of a set of two subchannels in the array and for other locations the significant number is less than seven which is then the maximum number of nonzero elements which can possibly occur in a row. (See Fig. 2.1)



1: Central Location: 6 affecting cross-flows 7 nonzero coefficients in the corresponding row of the array A



2: Corner Location: 3 affecting cross-flows 4 nonzero coefficients in the corresponding row of the array A



3: Edge Location: 4 affecting cross-flows 5 nonzero coefficients in the corresponding row of the array A

FIG. 2.1: Affecting Cross-Flow for all Types of Subchannel Location

Fig. 2.2 shows the origin of the matrix for a simple case. Note that the element on the principal diagonal is always different from zero, since the cross-flow on the particular considered boundary cannot be identically equal to zero.

If w_{ij} represents the cross-flow from subchannel i to subchannel j and w_{ji} represents the cross-flow from subchannel j to subchannel i, and since the two cross-flows are identical, i.e.,

$$w_{ij} = w_{ji} \quad ,$$

one notes that the cross-flow coefficient matrix is therefore symmetric. (See Fig. 2.3)

2.2 Width of the Band

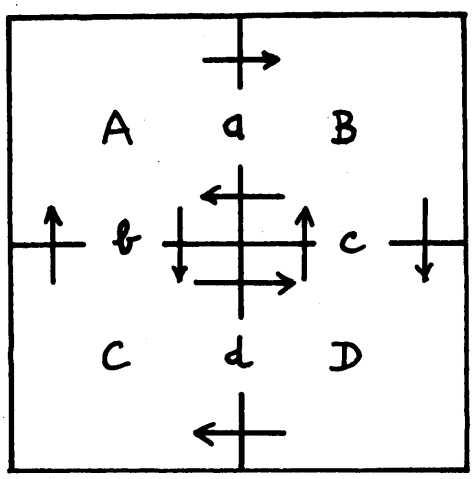
The location of the significant cross-flow coefficients within the row depends on the channel boundary numbering. As shown in Fig. 2.3, if the boundaries are arbitrarily numbered, the place of the cross-flow coefficient is arbitrary within each row of the matrix A.

Therefore, for computer applications, the identification of the significant elements can be a lengthy procedure. It is better to adopt a consistent boundary numbering scheme for the array: in this way, the coefficients can be located within a "zone" around the principal diagonal. (See Table 2.1)

A second step is to minimize the width of each zone in each row: the best technique consists of numbering the boundaries from the left to the right and from the top to the bottom in the array. [2]

Since the width of each zone is different from row to row, the last step consists of finding an overall envelope around the principal diagonal and containing every zone: in this way all significant cross-flow coefficients will be found within the envelope, also called the band.

In order to find the width of this band, one has just to compute the maximum value, among all the rows of the matrix, of the interval



For this case: nber of cross-flow = 4 $\Rightarrow$ A is a (4 × 4) array

cross-flow: a affected by b and c

cross-flow: b affected by a and d

cross-flow: c affected by a and d

cross-flow: d affected by b and c

cross-flow ↓ / eff. Boundary →	a	b	c	d
a	X	X	X	O
b	X	X	O	X
c	X	O	X	X
d	O	X	X	X

FIG. 2.2: Origin of Matrix A for the 4-Subchannel Case

TABLE 2 .1 : Typical AAA Matrix for a 10 Subchannel Case

	1	2	3	4	5	6	7	8	9	10	11	12
1	13.2	-6.4		6.4	-6.4							
2	-6.4	13.2	-6.4		6.4	-6.4						
3		-6.4	19.6			6.4						
4	6.4			13.2			-6.4		-6.4			
5	-6.4	6.4			13.2		6.4	-6.4		-6.4		
6		-6.4	6.4			19.6		12.8				
7				-6.4	6.40		13.2	-6.4	6.4	-6.4		
8					-6.4	12.8	-6.4	19.6		6.4		
9				-6.4			6.4		13.2		-6.4	-6.4
10					-6.4		-6.4	6.4		19.6	12.8	
11									-6.4	12.8	19.6	6.4
12									-6.4		6.4	19.6

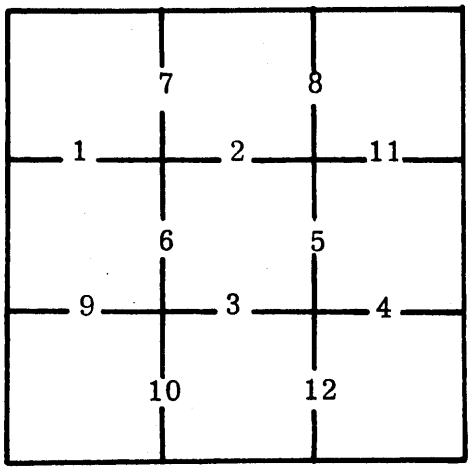


FIG. 2.3: Arbitrary Numbering Scheme for a 9-Subchannel Case

TABLE 2.2: (12 12) Array for an Arbitrary Numbering Scheme of a 9 Subchannel Case

	1	2	3	4	5	6	7	8	9	10	11	12
1	x	o	o	o	o	x	x	o	x	o	o	o
2	o	x	x	o	x	x	x	x	o	o	o	o
3	o	x	x	o	x	x	o	o	o	x	o	x
4	o	o	o	x	x	o	o	o	o	o	x	x
5	o	x	x	x	x	x	o	o	o	o	x	o
6	x	x	x	o	x	x	o	o	x	o	o	o
7	x	x	o	o	o	o	x	x	o	o	o	o
8	o	x	o	o	o	o	x	x	o	o	x	o
9	x	o	o	o	o	x	o	o	x	x	o	o
10	o	o	x	o	o	o	o	o	x	x	o	x
11	o	o	o	x	x	o	o	x	o	o	x	o
12	o	o	x	x	o	o	o	o	o	x	o	x

between the furthest element in each row and the corresponding element on the principal diagonal, and then to define the width of the band as being equal to twice this maximum value, since the matrix A is symmetric. (See Table 2.2)

2.3 General Remarks

The density of a matrix is defined as the fraction of nonzero elements. This has been computed in Table 2.3 for representative values of n, showing those which are more likely to be encountered in practice for the cross-flow problem.

It is known that for a very small value of density, as is the case in this problem, the Gauss-Siedel iterative method is more appropriate for solving such a linear system of equations. When programming the iterative method the sparsity of the matrix is used, since it is possible to code the matrix under a very compact form, and to generate it whenever it is needed.

For the general case of a $(n \times n)$ matrix, with a maximum of k nonzero elements per row, (k/n) being a very small quantity, it is possible to store this matrix in core memory under a $(k \times n)$ array, i.e. it is always possible to use a system of indices for which n times k places in the core memory are sufficient for the generation of A.

The coding used to generate A is discussed in Appendix B.

Note also that since $a_{ii} \neq 0$ for $(1 \leq i \leq n)$, an iterative process can be used. However, for the treatment of a general type of matrix, one must carefully check if all the diagonal elements are different from zero. If not, the matrix is singular and no unique solution exists: therefore, the computational time will be very large in this case since the solution never converges.

Table 2.2: Density of Different Sizes of Matrix A

Note: Density is defined by: $d = \frac{7\ n}{n^2} = \frac{7}{n}$

nber channels	10	128	193
nber boundaries	12	232	356
d	0.583	0.03017	0.01966

Finally, as noted before, the matrix A is symmetric. Then, it is seen that $A^T = A$, where A^T is the transpose of A: the eigenvalues are real.

It can be noted that the property of definiteness or semi-definiteness is discussed in Chapter 4 since this property greatly aids in the solution of linear system by almost any method.

In particular, if the definiteness property is quite irrelevant to solving the eigenvalue problem, it gives information on the sign of eigenvalues. If A is positive semi-definite, the eigenvalues are all non-negative and if A is positive-definite they are all positive. (6)

CHAPTER 3

INTRODUCTION TO GENERAL ITERATIVE PROCEDURES

3.1 Methods
3.1.1 Algebraic Formulation
3.1.1.1 General Case

In this section A is assumed to be a full matrix. It is also assumed that A $\in$ L (R^n) and is not singular in the vector equation.

$$A \underline{x} = \underline{y} \tag{3.1}$$

One of the simplest iterative methods is that of JACOBI.[9] As $a_{ii} \neq 0$, ($i = 1, \ldots, n$), it is possible to write for the first iterative solution to Eq. 3.1 as

$$x_1 = \frac{1}{a_{11}} \left(Y_1 - \sum_{j \neq 1}^{n} a_{1j} x_j \right) \tag{3.2}$$

If, as initial estimates, every x_j is chosen to be zero, the first iterate value of x_1, is

$$x_1 = \frac{Y_1}{a_{11}} \tag{3.3}$$

then for the second equation, x_2 is found to be:

$$x_2 = \frac{1}{a_{22}} \left(Y_2 - \sum_{\substack{j \neq 2 \\ j = 1, \ldots, n}}^{n} a_{2j} x_j \right) \tag{3.4}$$

or

$$x_2 = \frac{1}{a_{22}} \left(Y_2 - a_{21} x_1 \right) \tag{3.5}$$

In a similar way the ith component of the n-column vector $\underline{X}$ is obtained by:

$$x_i = \frac{1}{a_{ii}} \left(Y_i - \sum_{j \neq i} a_{ij} x_j \right) \qquad (3.6)$$

At the next iteration every term x_i ($i = 1, \ldots, n$) is then known and the iterative process is applied again.

If (k) is the superscript of the kth iteration

$$x_1^{(k)} = \frac{1}{a_{11}} \left(Y_1 - \sum_{j \neq 1}^{n} a_{1j} x_j^{(k-1)} \right) \qquad (3.7)$$

then the ith component of the n-column vector $\underline{X}$ is given by:

$$x_i^{(k)} = \frac{1}{a_{ii}} \left(Y_i - \sum_{j \neq i} a_{ij} x_j^{(k-1)} \right) \qquad (3.8)$$

with $i = 1, 2, \text{----}n$

Jacobi's method can be improved by the following considerations:

Assuming the computations of Eq. 3.8 are done sequentially for i = 1, 2, ---n, then for computing the kth iteration of the ith component, $x_i^{(k)}$, it is advised to use the "new" components, $a_j^{(k)}$, with (j = 1, ---, i -1), which are then available instead of the "old" components, $x_1^{(k-1)}$, with (j = 1, ---, i -1); . Of course, the other components $x_2^{(k)}$, with l = i + 1, ---, n, are not yet known. The components $x_2^{(k-1)}$, with l = i + 1, --n are used to compute $x_i^{(k)}$, whose expression is now:

$$x_i^{(k)} = \frac{1}{a_{ii}} \left(Y_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} \right) \qquad (3.9)$$

This is the Gauss-Siedel [6, 8] iterative method, which has been used to solve the cross-flow problem, and has been implemented as explained in Chapter 4, to improve its rate of convergence.

Note that Appendix B explains the program developed from this iterative method.

3.1.1.2 Case of the Cross-Flow Coefficients Matrix

Two interesting remarks concerning the computational problem of the Gauss-Siedel method applied to the cross-flows matrix A, can be made.

1) The matrix is band striped: outside the band the upper and lower triangular parts contain only zero elements. No operation should then be performed in these two zones in order to avoid needless computational expense.
2) The sparsity of A implies that operations should only be made with significant coefficients in the row, for the same reason.

These two remarks are closely related to the problem of the storage of the matrix A in a compact array, with a system of indices allowing one to generate it whenever needed, i.e. for the computation of the basic iterative equation (3.9).

In this way, one minimizes the computation time of this solution method, by allowing only operations involving the nonzero values of the coefficients, in the row, and the corresponding cross-flows components of the cross-flow vector.

For all these reasons, the generation of matrix A will be given first consideration in the program.

3.1.2 Matrix Formulation [6, 7]

It may prove useful to derive the Gauss-Siedel iterative method under the matricial form:

If $D = \text{diag}(a_{11}, \ldots\ldots, a_{nn})$ and

$B = D - A$

Eq. 3.8 becomes

$$x_i^{(k)} = D^{-1} B \underline{x}^{(k-1)} + D^{-1} Y_i \qquad (3.10)$$

Now, let (-E) be the strictly lower triangular part of A and (-F) the strictly upper triangular part; that is,

$$E = \begin{bmatrix} 0 & & \\ a_{21} & \ddots & 0 \\ \vdots & & \\ a_{n1} & a_{n,n-1} & 0 \end{bmatrix} \qquad (3.11)$$

$$F = \begin{bmatrix} 0 & a_{21} & & a_{1n} \\ & \ddots & & \vdots \\ & & \ddots & a_{n-1,n} \\ 0 & & & 0 \end{bmatrix} \qquad (3.12)$$

then, with $D = \text{diag}(a_{11}, \ldots\ldots, a_{nn})$, the matrix A becomes:

$$A = D - E - F \qquad (3.13)$$

and Eq. 3.9 is equivalent to:

$$D \underline{x}^{(k)} = \underline{y} + E \underline{x}^{(k)} + F \underline{x}^{(k-1)}. \qquad (3.14)$$

The fact that $a_{ii} \neq 0$, $i = 1, \ldots\ldots, n$ insures that (D - E) is non-singular and the Gauss-Siedel iteration can be written as:

$$\underline{x}^{(k)} = (D - E)^{-1} F \underline{x}^{(k-1)} + (D - E)^{-1} \underline{y}, \qquad (3.15)$$

$k = 0, 1, 2, \ldots\ldots$

Now with $\begin{cases} H = (D - E)^{-1} F \\ \underline{d} = (D - E)^{-1} \underline{y} \end{cases}$

Eq. 3.15 becomes:

$$\underline{x}^{(k)} = H \underline{x}^{(k-1)} + \underline{d}. \qquad (3.16)$$

Then if $\underline{x}^*$ represents the solution, it is clear that: $\underline{x}^* = H\underline{x}^* + \underline{d}$ only if $A\underline{x}^* = \underline{b}$ (3-16 bis)

3.2 Convergence Theorem for Linear Iterative Methods

Before introducing the concept of convergence for linear iterative methods, it is worthwhile to review the following useful properties.

Let $L(R^n)$ represent the set of real $(n \times n)$ matrices, as is the case in this problem. As $A \in L(R^n)$ the following properties are equivalent.

Theorems:
a) A is non-singular
b) det A = 0; (i.e. the linear system $A\underline{x} = 0$ has only the solution x = 0)
c) for any vector b, the linear system $A\underline{x} = \underline{b}$ has a unique solution [6]

If $A \in L(C^n)$, where C^n is the set of complex $(n \times n)$ matrices, then a scalar (real or complex) and a vector $\underline{x} \neq 0$ are eigenvalue and eigenvector of A if

$$A \underline{x} = \lambda \underline{x}, \qquad (3.17)$$

or if λ is eigenvalue of A

$$\det (A - \lambda I) = 0. \qquad (3.18)$$

Then A has n (not necessarily distinct) eigenvalues which are the n roots of Eq. 3.13.

The collection of these n eigenvalues $\lambda_1, \ldots, \lambda_n$ is called the spectrum of A and

$$\rho(A) = \max_{1 \leq i \leq n} |\lambda_i|$$

is the spectral radius of A.

This information is useful for improving the rate of convergence of the method as explained in Chapter 4.

Coming back to the theorems; let $H \in L(R^n)$ and assume that the equation

$$\underline{x} = H\underline{x} + \underline{d} \tag{3.19}$$

has a unique solution $\underline{x}^*$. Then Eq. 3.16 converges to $\underline{x}^*$ for any initial guess $\underline{x}^0$ if and only if $\rho(H) < 1$.

Subtracting

$$\underline{x}^* = H\underline{x}^* + \underline{d}, \tag{3.20}$$

from Eq. 3.16, the equation representing the "error" of the solution is obtained:

$$\underline{x}^{(k+1)} - \underline{x}^* = H(\underline{x}^{(k)} - \underline{x}^*) = H^{(1)}(\underline{x}^{(k-1)} - \underline{x}^*)$$

$$= \ldots\ldots\ldots = H^{(k+1)}(\underline{x} - \underline{x}^*) \tag{3.21}$$

Thereby, in order that

$$\lim (\underline{x}^{(k)} - \underline{x}^*) = 0 \quad \text{for any}$$

$\underline{x}^0$ it is necessary and sufficient that [13]

$$\lim_{k \to \infty} H^{(k)} = 0$$

This is true, if $\rho(H)$ is inferior to 1. It is then possible to choose a norm on R^n such that $\|H\| < 1$.

Then, $\|H^{(k)}\| \leq \|H\|^{(k)} \longrightarrow 0$ as $k \longrightarrow \infty$

As a result, the convergence of Eq. 3.16 is reduced by this theorem, to the algebraic problem of showing that

$$\rho(H) < 1$$

CHAPTER 4
IMPROVEMENT OF AN ITERATIVE TECHNIQUE

4.1 Initial Value Problem

In a convergent iterative process, the choice of the initial value, i.e. $\underline{x}^0$, is not of primary importance: the accuracy of the solution which can be obtained is not determined by this choice.

However, in the case of this problem, the first set of assumptions has to be estimated carefully since the cross-flows solution is carried out several times in the course of a program run: once for the steady state case and seven times (in seven outer iterations) for a transient state case.

For the steady state case, once the parameters involved in the cross-flow problem are known, the remaining task is the solution of the matrix equation. After a certain number of iterations for which the convergence criteria is satisfied (see section 4.2) one obtains a cross-flow vector for each axial step.

This information is kept in the core-memory, and is used later as the initial value for the first calculation step of the transient case.

This procedure is justified by the saving of computation time for carrying out the calculation of the transient case: once a selected type of transient and its corresponding code inputs are designed, the parameters involved in the set-up of the cross-flow problem change, and one ends up with a different coefficient matrix A and therefore with a different cross-flows solution at each axial step.

The objective of the calculation is the knowledge of the change of every thermal and hydraulic parameter, in the very first seconds of

the transient; therefore, the code is run seven times for a careful determination of these evolving parameters during the seven first periods of time of the transient.

Therefore, the cross-flows solution corresponding to the steady state case comprises the initial guess for the first time step of the transient solution. Generally, the cross-flows solution at time (t - 1) is used as the initial guess of the calculation at time (t).

4.2 Convergence Criteria, Norms

A number of convergence criteria were used experimentally in the work. The most complete convergence criterion associated with the Gauss-Siedel Technique was found to be that of the comparison of the relative change in the approximation to each of the unknown cross-flows to a given error criterion [5];

that is

$$\left| \frac{x_i^{(k)} - x_i^{(k-1)}}{x_i^{(k)}} \right| \leq \varepsilon \qquad (4.1)$$

for all $i = 1, \ldots, n$
at each axial step J.

However, this convergence criterion is found to be too strict for two reasons. First, if in the same axial step, the different cross-flow components have different orders of magnitude, the number of required iterations may become too large for economical solutions.

Also, for two different axial steps, the order of magnitude of the solution, being defined here as an "average" of the absolute values of the cross-flow components, may be different.

In order to save computer processing unit (CPU) time, it has been suggested that one should use a different and relative convergence criterion ε' for each axial step.

Any type of norm contains the necessary information regarding the magnitude of the average cross-flow vector, to be an acceptable "weighting" parameter.

The three types of norms which could be used are:

1. $\| \underline{x} \|_E = \left(\sum_{i=1}^{n} | x_i |^2 \right)^{1/2}$ Euclidian norm,

2. $\| \underline{x} \|_S = \sum_{i=1}^{n} | x_i |$ sum norm, and

3. $\| \underline{x} \|_\infty = \max_{1 \leq i \leq n} | x_i |$ max norm.

The max norm has not been selected as the scale factor for the axially-varying ε because of the calculation time it would have required at each axial step during each time step.

The sum norm has not been used under this form, in the computer code related problem, but as an average value coupled with exterior value of the flow in the rod bundle as explained later in section 4.4.

The Euclidian norm is the right one for its simplicity and adequacy in the computational problem. (See Appendix B)

It is possible now to define a relative criterion ε', being the product of a fixed arbitrary constant, ε, by the Euclidian norm of the cross-flow vector for the given axial step. This way the convergency criterion weighted by the order of magnitude of the cross-flow norm.

Relation (4.1) becomes

$$| x_i^{(k)} - x_i^{(k-1)} | \leq \varepsilon' = \varepsilon * \| \underline{x}^{(k)} \|_E \tag{4.2}$$

The value of ε is chosen after a set of numerical experiments to be equal to: 10^{-3}.

The objective of this change is to reduce the calculation time necessary to compute the component of the cross-flow vector which has the smallest order of magnitude. One can simply say that if the order of magnitude of the cross-flow solution is 10^{-1}, in a particular axial step ℓ if by chance one of the cross-flow components is much smaller, i.e. 10^{-3}, one is not interested in the precision of this cross-flow estimate, since it will not affect significantly the accuracy of the overall cross-flow mass balance in the axial plane.

4.3 Improving the Rate of Convergence of the Gauss-Siedel Method

4.3.1 Successive Over Relaxation

Using the following notation [9] one can write an expression for the change between iterates in the Gauss-Siedel method

$$\underline{x}^{k+1} = \underline{x}^{k} + \underline{r}^{k}, \tag{4.3}$$

with

$$\underline{r}^{k} = (D - E)^{-1} (A \underline{x}^{(k)} - \underline{y}). \tag{4.4}$$

Generally, an iteration algorithm can be written under the form of Eq. 4.3, with different $\underline{r}^k$ for different algorithms.

Now, in order to accelerate the convergence of the iteration, one can multiply the residual vector by a real number Ω such as: $1 < \Omega < 2$, with the result

$$\underline{x}^{k+1} = \underline{x}^{k} + \underline{r}^{k}, \text{ and} \tag{4.5}$$

the residual vector being given as

$$\underline{r}^k = D^{-1} E \underline{x}^{(k+1)} + D^{-1} F \underline{x}^{(k)} + D^{-1} \underline{y} - \underline{x}^k. \qquad (4.6)$$

Eq. 4.5 takes the form

$$\underline{x}^{(k+1)} = \underline{x}^k + \Omega \left[D^{-1} E \underline{x}^{(k+1)} + D^{-1} F \underline{x}^{(k)} + D^{-1} \underline{y} - \underline{x}^{(k)} \right] \qquad (4.7)$$

or

$$\underline{x}^{(k+1)} = \underline{x}^k \left[1 - \Omega\right] + \Omega \left[D^{-1} E \underline{x}^{(k+1)} + D^{-1} F \underline{x}^{(k)} + D^{-1} \underline{y} \right] \qquad (4.8)$$

Using the notation

$$\underline{x}^{(k+1)} = D^{-1} E \underline{x}^{(k+1)} + D^{-1} F \underline{x}^{(k)} + D^{-1} \underline{y} \qquad (4.9)$$

which represents the result in the Gauss-Siedel iteration; Eq. 4.5 becomes

$$\underline{\tilde{x}}^{k+1} = \underline{x}^k \left[1 - \Omega\right] + \Omega \underline{x}^{(k+1)}. \qquad (4.10)$$

4.3.2 Convergence Criteria Considerations for SOR

A necessary and sufficient condition for the convergence of any iterative algorithm is that the spectral radius of the iteration matrix be less than unity. (12, 13)

In the case of the Gauss-Siedel method the necessary conditions which insure convergence are positive definiteness and diagonal dominance. (10, 11)

The following definitions (10) are of interest in this regard:

1. An $(n \times n)$ matrix $A = (a_{ij})$ is weakly diagonally dominant by rows if

$$|a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|$$

for all $1 \leq i \leq n$, with strict inequality required for at least one value of i, the matrix is strictly diagonally dominant if strict inequality holds for all i, similar conditions hold for diagonal dominance by columns;

The concept of scaling can be introduced by the following definition:[10]

Let A be an $(n \times n)$ matrix, then if there exists a scaling of the columns [rows] of A by a set of nonzero multipliers such that the transformed matrix is strictly diagonally dominant by rows [columns], then A possesses generalized diagonal dominance by rows [columns];

3. Let $A = (a_{ij})$ and $B = (b_{ij})$ be two $n \times r$ matrices. Then $A \geqslant B$ if $a_{ij} \geqslant b_{ij}$ for all $1 \leq i \leq n$, $1 \leq j \leq r$. If 0 is the null matrix and $A \geqslant 0$, then A is a non-negative matrix. Similar conditions hold for strict inequality, and if $A > 0$, then A is a positive matrix.

With these definitions, one may outline a convergence criterion for general matrices. The following theorem shows that a general coefficient matrix may be scaled arbitrarily by rows and by columns without affecting the asymptotic convergence, the successive over relaxation.

The restrictions for the properties of the matrix are also presented:[10]

Let $A = D + E + F$ be an $(n \times n)$ coefficient matrix where $D = \text{diag}(a_{11}, \ldots, a_{nn})$, $a_{ii} \neq 0$ for $1 \leq i \leq n$ and E, F are respectively strictly lower and upper triangular matrices. Let a matrice A be obtained from A by scaling the rows and columns of A with arbitrary nonzero multipliers. Then the SOR iteration matrices for A and A' will have the same eigenvalues.

The SOR iteration matrix for A is given as

$$M_{\Omega} = (D + \Omega E)^{-1} \left[(1-\Omega) D + \Omega F \right] , \text{ and}$$

by construction,

$$A' = P A Q,$$

where P and Q are diagonal matrices with nonzero diagonal entries. The diagonal, lower triangular and upper triangular parts of A' are respectively given as

$$PDQ,\ PEQ,\ PFQ,$$

and the SOR iteration matrix for A' is therefore seen to be

$$M'_{\omega} = \left[P(D + \Omega E) Q \right]^{-1} \left[(1-\Omega) PDQ - \Omega PFQ \right]$$

$$= Q^{-1} M_{\omega} Q .$$

thus M and M'w have the same eigenvalues.

From this theorem, useful corollaries are decided, which would be applied to the problem.(10)

Corollary 1: The Successive over Relaxation (SOR) method is convergent for A if and only if it is convergent for A', and the asymptotic rates of convergence are the same in both cases.

Corollary 2: If scaling by rows or columns can produce a matrice A' which is diagonally dominant then the Gauss-Siedel and Jacobi methods are convergent for the original matrix A.

Corollary 3: If the transformed matrix A' is diagonally dominant, then, Successive over Relaxation (SOR) is guaranteed to converge for Ω in the range.

$$0 < \Omega < \Omega' = \frac{2}{1 + \min (S_1, S_2)} ,$$

with $S_1 = \max_i \dfrac{\sum_{j \neq i} |a'_{ij}|}{|a'_{ii}|}$,

$$S_2 = \max_j \frac{\sum_{i \neq j} |a'_{ij}|}{|a'_{ii}|} .$$

It follows that Successive over Relaxation (SOR) is convergent also for A in the same range.

For the validity of the experimental value of Ω found for Successive over Relaxation (SOR), in the COBRA IIIC cases examined in this work, (see Appendix B), a verification could be done in the range $0 < \Omega < \Omega'$, such as is discussed above to insure convergence of the method for a general coefficients matrix.

Finally, concerning The definiteness it should be interesting to note that Ostrowski[13] proved that for a coefficient matrix with positive diagonal element a necessary and sufficient condition for the convergence of Successive over Relaxation (SOR) is that the matrix be positive definite.

These remarks are intended to give a better understanding of all the transformations which can be performed to convert A to a positive diagonal dominant matrix, and to include in the general computational treatment of the problem, a routine for checking the convergence of A', before undertaking the solution scheme.

CHAPTER 5
FURTHER MODIFICATIONS OF THE METHOD

5.1 Finding the maximum eigenvalue: Iterative Refinement

The calculation of the eigenvalue spectrum has not been considered in this work, since it would have affected the overall computational time. However, one can use the maximum value of the eigenvalue spectrum to hasten the convergence of the process, without spending additional computational time.

Using at each iteration the vector $\underline{\Delta}^k$ such as:

$$\underline{\Delta}^k = \underline{X}^k - \underline{X}^{k-1} \tag{5.1}$$

It is known that for large value of k the ratio

$$\lim_{k \to \infty} \left(\frac{\Delta_i^k}{\Delta_i^{k-1}} \right) = \lambda_1 \tag{5.2}$$

for any i = 1, 2, ---n

This relation could be used as a kind of convergence criterion: If for the iterations k - 1, k, k + 1, the ratios

$$\left(\frac{\Delta_i^k}{\Delta_i^{k-1}} \right) \simeq \left(\frac{\Delta_i^{k+1}}{\Delta_i^{k}} \right) \tag{5.3}$$

are approximately equal, one can consider that the solution has been obtained and stops the iteration.

The value λ_1 can be given a certain range of uncertainty within which the value of the ratio, of two successive iterations, can

be considered equal to λ_1 .

The final solution is then

$$\underline{X}^{\infty} = \underline{X}^{k} + \frac{\underline{\Delta}^{k}}{1 - \lambda_1} \tag{5.4}$$

since:

$$\underline{X}^{k+1} = \underline{X}^{k} + \underline{\Delta}^{k}$$

$$\underline{X}^{k+2} = \underline{X}^{k} + \underline{\Delta}^{k} + \underline{\Delta}^{k+1}$$

$$\vdots$$

$$\underline{X}^{\infty} = \underline{X}^{k} + \underline{\Delta}^{k} (1 + \lambda_1 + \lambda_2 + \text{------}) \tag{5.5}$$

and

$$\sum_{i=0}^{\infty} \lambda_k^{i} = \frac{1}{1 - \lambda_k} \tag{5.6}$$

Note also that when λ_1 is known, one can compute Eq. 5.4 for each component i, i.e.

$$X_i^{\infty} = \underline{X}_1^{k} + \frac{\Delta_i^{k}}{1 - \lambda_1}$$

5.2 Other Possible Values for Convergence Criterion

As mentioned in Section 4.2, the choice of an adequate convergence criterion is important. If it is chosen to be too strict, large computational requirements can occur, and the results will not be significant: some of them will have an order of magnitude which will not even affect the cross-flow average value in a given axial plane.

If it is chosen to be too loose, the results will not be sufficiently accurate.

By choosing the norm of the cross-flow vector in each axial plane, one is interested in obtaining an average value of the order of magnitude of the cross-flows.

However, other different convergence criteria related to an exterior parameter have been examined. For example, the average axial flow rate value in the core which is previously calculated, is used as a reference "parameter" and the cross-flow convergence criterion was taken to be (10^{-3}) times this value.

As explained in detail in Appendix B different parameters such as FLO, FERROR, MOYFLO have been examined.

5.3 Sensitivity Analysis: Finding Solution by Components

An interesting refinement of the method could now be considered. As explained in Section 4.3 the cross-flow vector is computed in such a way that the inner iterations end when each of the cross-flow components satisfies the convergence criterion. One can remark that since the magnitudes of the various cross-flow components are not necessarily uniform some of them will have satisfied the convergence criterion from early in the calculation while others will not yet have done so. The time required to complete all of the required calculations, namely, the inner iterations, for all components, can be very large for large order matrices.

If and only if the cross-flow components converge uniformly and monotonically, the iteration procedure can be modified in such a way that after one cross-flow component has reached its solution, namely, after having satisfied a given convergency criterion, the outer iteration, i.e. calculation of cross-flow vector, will not change nor modify this value.

One can note the considerable calculation saving time which could be obtained for large cross-flow vectors.

However, one must be careful when using this technique: a sensitivity analysis, for a typical small case, should be carried out in order to estimate the effect of the truncation error on the cross-flow components. In Appendix B a complete set of instructions is proposed for a print-out of COBRA III C results at different typical stages of calculation.

CHAPTER 6

CONCLUSION AND DISCUSSION OF RESULTS

The superiority of an iterative technique over a direct solution method, in the particular case of the large sparse matrix problem, is not obvious. However, it can be made plausible by the following remarks.

Remark 1: Two possible types of matrices can be handled throughout the introbundle cross-flows calculation. In one type the cross-flow coefficient matrix may be diagonally dominant: in this case and for any size of matrix the rate of convergence is rapid and an iterative technique, like the Successive over Relaxation (SOR) used in this work will be as much as four times quicker than the Gauss Elimination, according to other numerical experiments [15] and therefore will show its clear superiority in calculation time. In the other case, if the matrix is not diagonally dominant, the rate of convergence will be slower and therefore a breakeven matrix size can be found when comparing computation times required by the direct method (e. g. Gauss Elimination) and those needed by the Successive over Relaxation method. The difficulty in judging the merit of one or the other method arises when the accuracy of the results must be taken into consideration.

Remark 2: When one compares the results given by the direct method to the results given by the iterative technique one notes that:

a) If the results are identical up to the fifth decimal - and omitting the round-off error, - the calculation time is larger (approximately twice as large) for the iterative method when compared to those given by the Gauss

Elimination for small order matrices (12 × 12). For large order matrices (128 × 128) the calculation time is faster by 35% for the iterative technique. An estimate - based on numerical experiments and an empirical correlation proposed by R. W. Bowring [3] - predicts an 80% advantage to the iterative method over the Gauss Elimination for very large order matrices (360 × 360). Note that these experiments verify approximately the theoretical estimates: $(\underline{n}^3 + n^2)$ operations are required when using the Gauss Elimination and $(2n^2 + n)$ are needed for each iteration when using the Gauss-Siedel method [5]. Since satisfactory results are obtained after an average of 14 iterations when using the iterative technique, one finds that the breakeven size appears at an approximate matrix size of 62. Thereafter, for larger order matrices the superiority of the iterative technique is confirmed. (See table 6.1)

b) As explained in Section 4.2 and 5.2, if one uses a convergence criterion equal to 10^{-3} in the relative convergence test of the iterative technique, one notes a significant time reduction in the calculation. When comparing the cross-flow results to those given by the Gauss Elimination, a discrepancy of the order of 10^{-3} appears in the results but this does not affect the mass flow rates and enthalpy results. One must recall that the Gauss Elimination is a direct method and therefore that it gives exact results. The great advantage of the iterative technique is its flexibility because of the adjustable value of the convergency criterion. Since a small discrepancy in the cross-flow results does not affect outside parameters, it is then worthwhile to loosen the convergency

Table 6.1: Overall Time Calculation Comparison between Iterative Technique and Gauss Elimination for Typical Cases Encountered in the Cross-Flow Problem

SIZE OF TYPICAL A MATRIX	DIAGONALLY DOMINANT MATRIX				NON-DIAGONALLY DOMINANT MATRIX			
	IDENTICAL RESULTS.		10^{-3} VARIATION		IDENTICAL RESULTS.		10^{-3} VARIATION	
	ITERATIVE TECHNIQUE	DIRECT METHOD	ITERATIVE TECHNIQUE	DIRECT METHOD	ITERATIVE TECHNIQUE	DIRECT METHOD	ITERATIVE TECHNIQUE	DIRECT METHOD
12 x 12 (Small order)	$T_1 = T_2/4$	T_2			$T_1 > T_2$ $T_1 = 1.5T_2$	T_2	$T_1 < T_2$ $T_1 = .95T_2$	T_2
62 x 62 (breakeven size)	$T_1 = T_2/4$	T_2			$T_1 = T_2$	T_2	$T_1 < T_2$ $T_1 = .85T_2$	T_2
128 x 128	$T_1 = T_2/4$	T_2			$T_1 = .65T_2$	T_2	$T_1 = .85T_2$	T_2
Large order up to (356 x 356)	$T_1 = T_2/4$	T_2			$T_1 = .20T_2$	T_2	$T_1 = .20T_2$	T_2

Note: 1. T_2 = Computation Time by Direct Method

2. Identical results mean except for round-off errors, results can be considered identical up to the fifth decimal.

criterion and get then a considerable saving in calculation time. Results for different typical matrices are reported in Table 6.1.

c) It is finally recalled that a value equal to 10^{-2} for the convergence criterion corresponds to an accuracy threshold value since for this value and larger value of the convergence criterion, the cross-flow results are very different from those obtained by the Gauss Elimination method and consequently the enthalpy and mass flow rates results are different from those found by the direct methods. For all these reasons and as explained in section b of Remark 2, 10^{-3} is by experiment the most appropriate value for the convergency criteria when the relative convergence test is used in the Iterative Technique.

Remark 3: Depending on the IBM Fortran computer level which is used differences in calculation time can occur. For example, if an IBM Fortran GI-level computer is used, two successive runs of a same problem show a 10% variation in calculation time. If an IBM Fortran H-level computer is used, the calculation time is approximately reduced by a factor of two and variation in calculation time for two successive runs is no more than 5%. (See Figs. 6.1, 6.2, 6.3)

Remark 4: Finally, a statement for judging the overall efficiency of the method will be based on the following formula proposed by R. W. Bowring [3]. The IBM 370/175 calculation time for the cross-flow problem with the Gauss Elimination method has been found to be given as:

$$T = 0.00282\ N K + 0.000\ 000\ 837\ N K \quad M S^2 + 0.0000\ 127\ NK^2.$$

The first term is the contribution to the calculation time for set-up of the matrix. In both methods - Gauss Elimination and Iterative

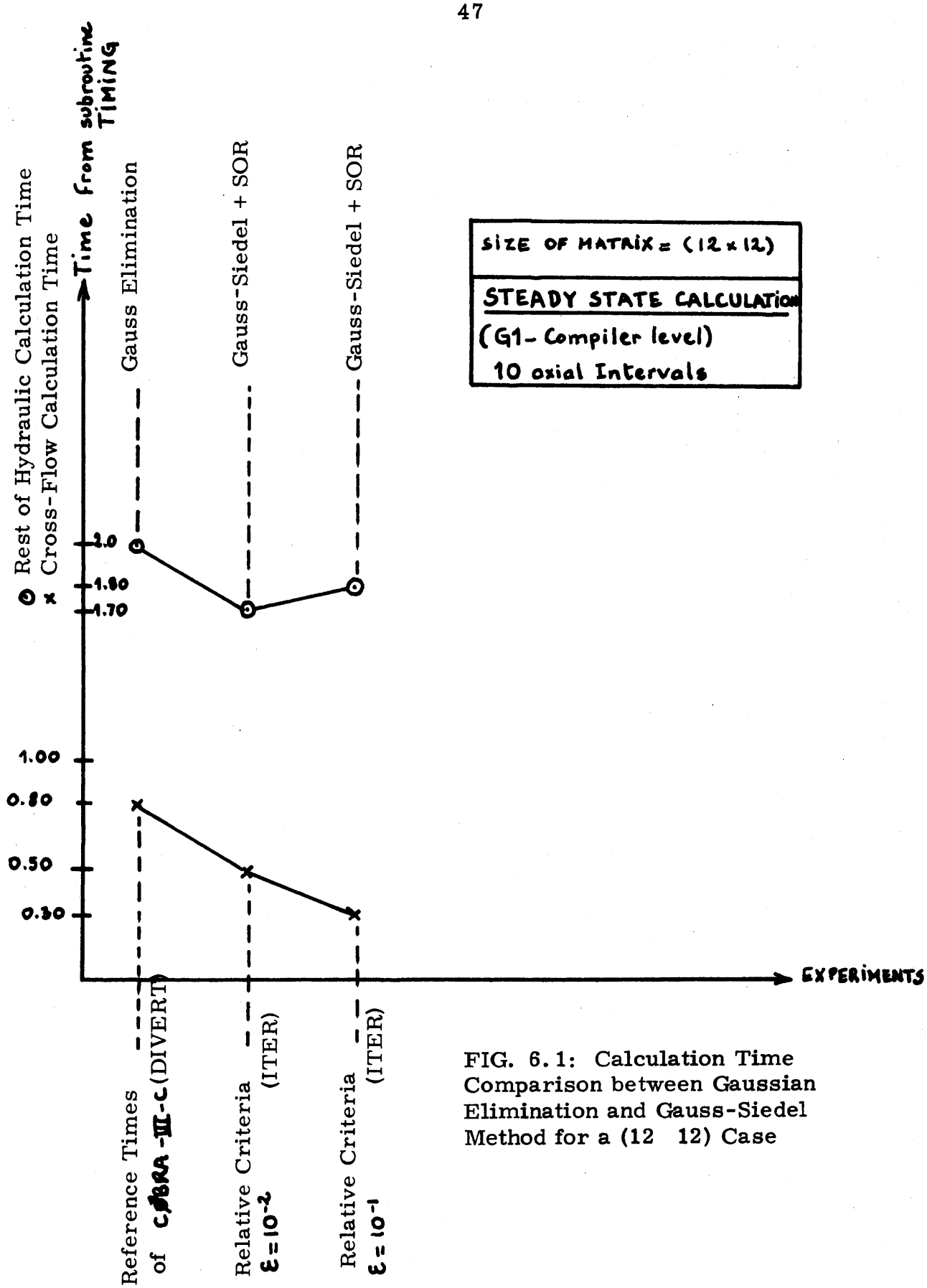


FIG. 6.1: Calculation Time Comparison between Gaussian Elimination and Gauss-Siedel Method for a (12 12) Case

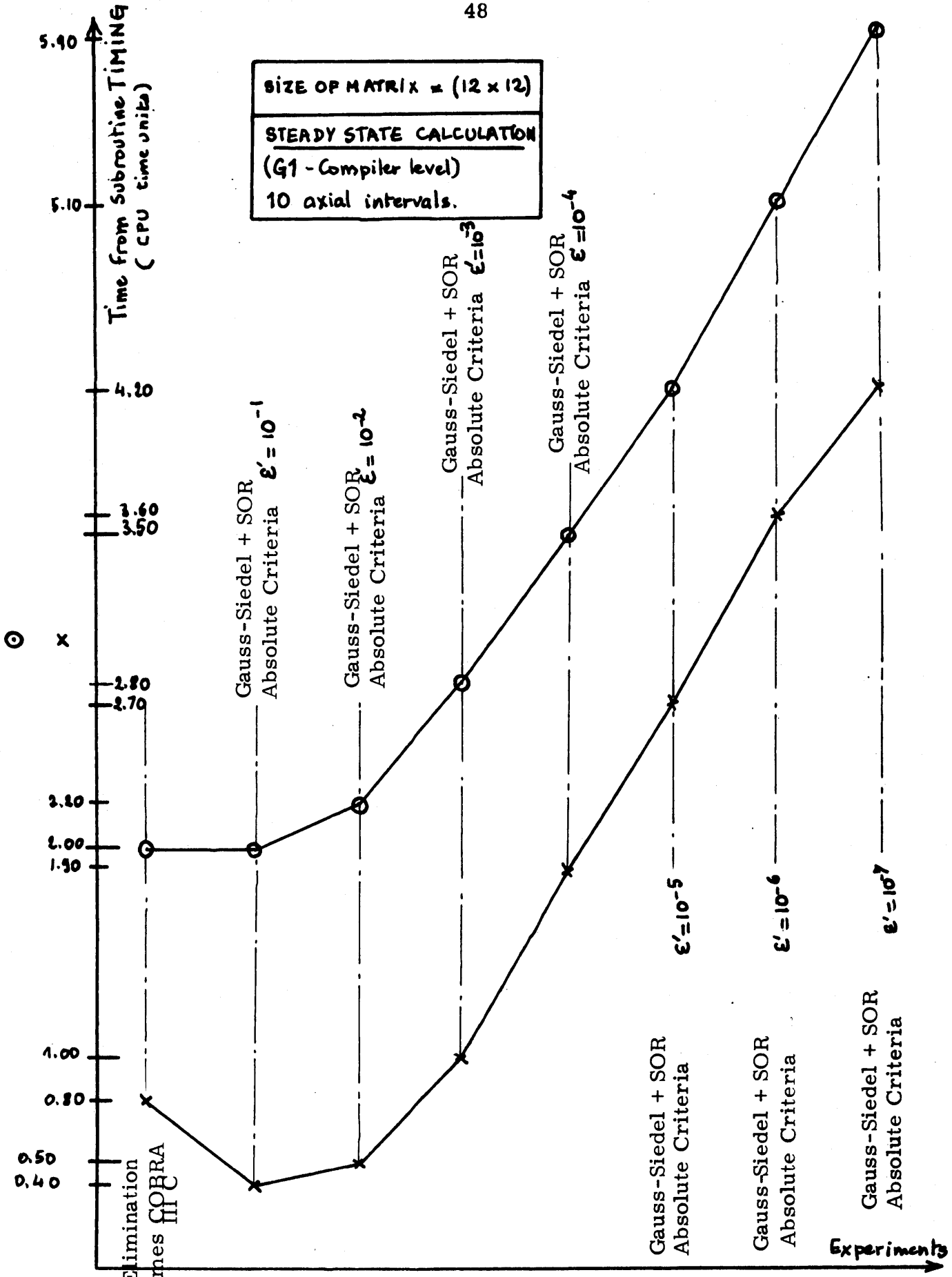


FIG. 6.2: Calculation Time Comparison Between Gaussian Elimination and Gauss-Siedel Method for Different Values of the Absolute and Relative Criterion

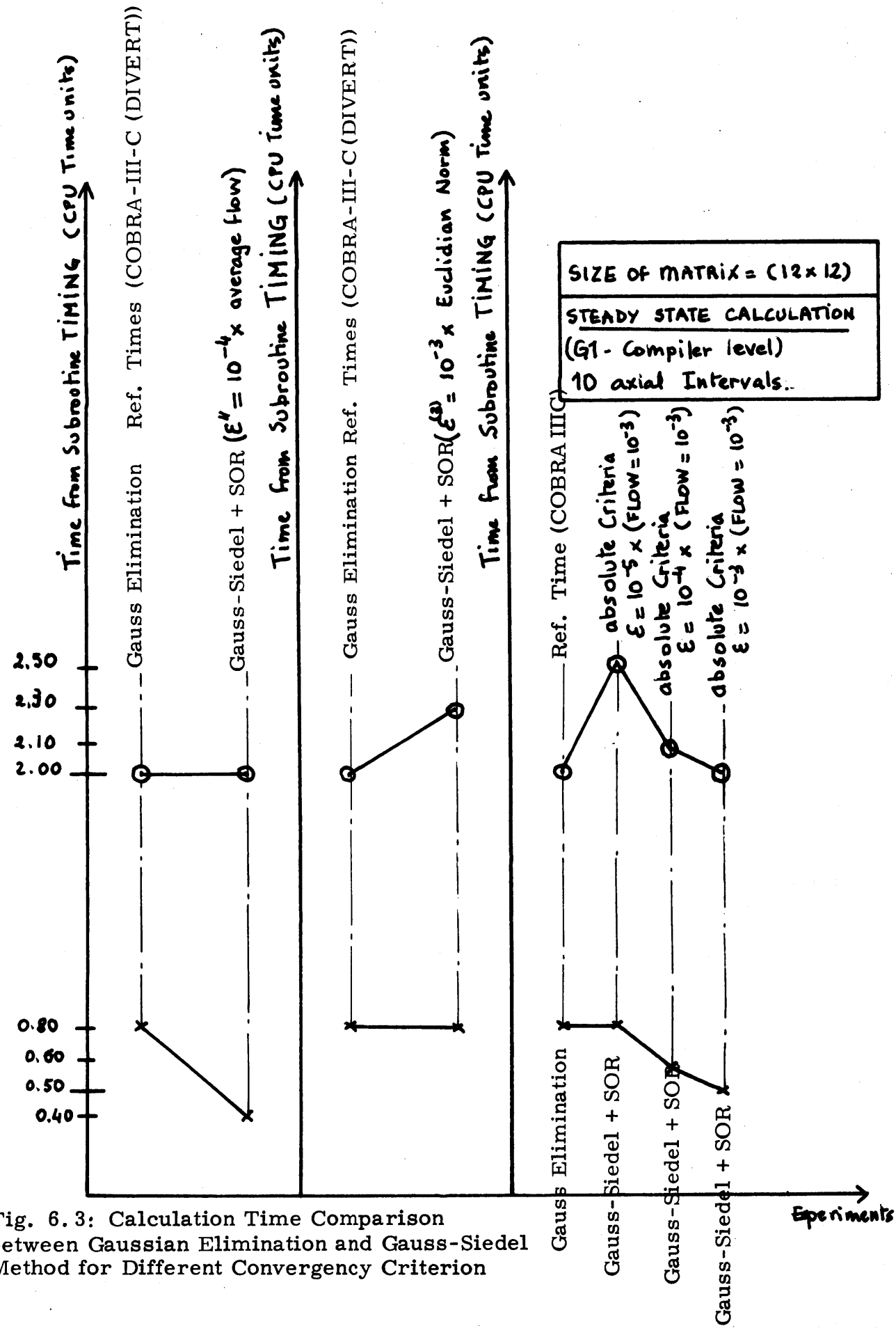


Fig. 6.3: Calculation Time Comparison between Gaussian Elimination and Gauss-Siedel Method for Different Convergency Criterion

Technique - it has the same value.

The second term represents the calculation time for storage of the coefficient matrix. As seen in Chapter 1 when using the Gauss Elimination method the value of the variable MS can vary from 11 to 63. For the iterative method the value of MS is fixed to 7. Therefore, there is a clear superiority in calculation time for the Iterative Technique over the Gauss Elimination method.

The last term concerns the cross-flow calculation time. To clarify the validity of the proposed formula one must recall that in Chapter 1 theoretical estimates of the required number of operations to be performed for both methods have been presented [5, 6]. It has been shown that ($\underline{n}^3 + n^2$) operations are required for the Gauss Elimination solution while ($2n^2 + n$) are needed for the Gauss-Siedel Iterative Technique. Therefore, without any false speculation one can deduce that the proposed formula underestimates the calculation time required by the Gauss Elimination: this time must be at least proportional to N^3 while those for the Gauss-Siedel method must be proportional to N^2. (See Figs. 6.4, 6.5)

From this remark it is clear that the formula should be modified to predict a more realistic time estimate from the Gauss Elimination method for large order matrices.

The overall conclusion is stronger after the above explanations: the Successive over Relaxation method has proved to be practically comparable to the Gauss Elimination for small order matrices (12 12) but is considerably superior in terms of computational effort when large order matrices are considered.

STEADY STATE CALCULATION (from MEK-31, Table 2)		
G1-Compiler level	H-Compiler level	G1-Compiler level
Set-up from DIVERT (COBRA-III-C) ①	Set-up from DIVERT (COBRA-III-C) ②	Set-up from ITER (COBRA-III-C MIT) ③
10 axial steps	10 axial steps	10 axial steps

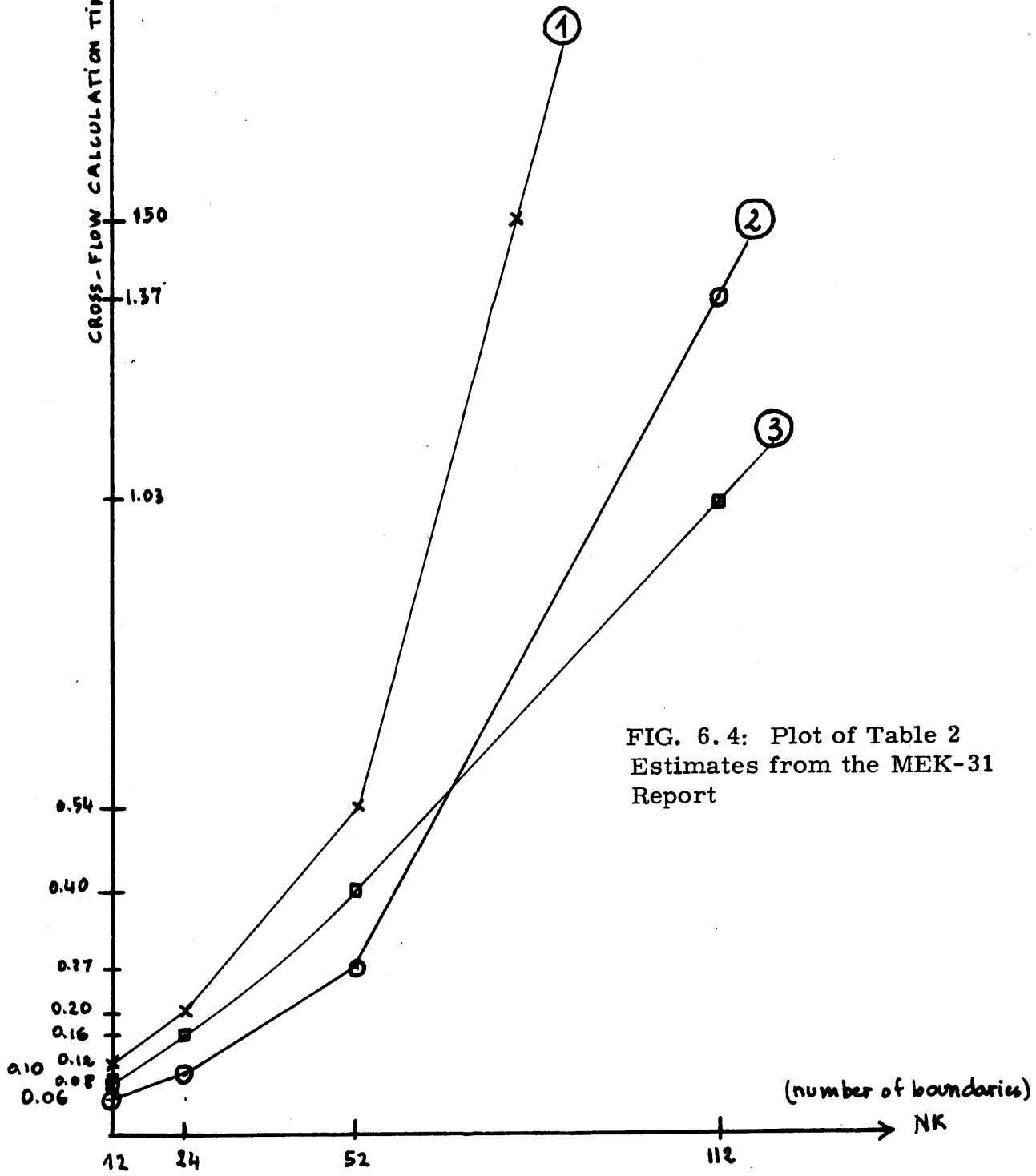


FIG. 6.4: Plot of Table 2 Estimates from the MEK-31 Report

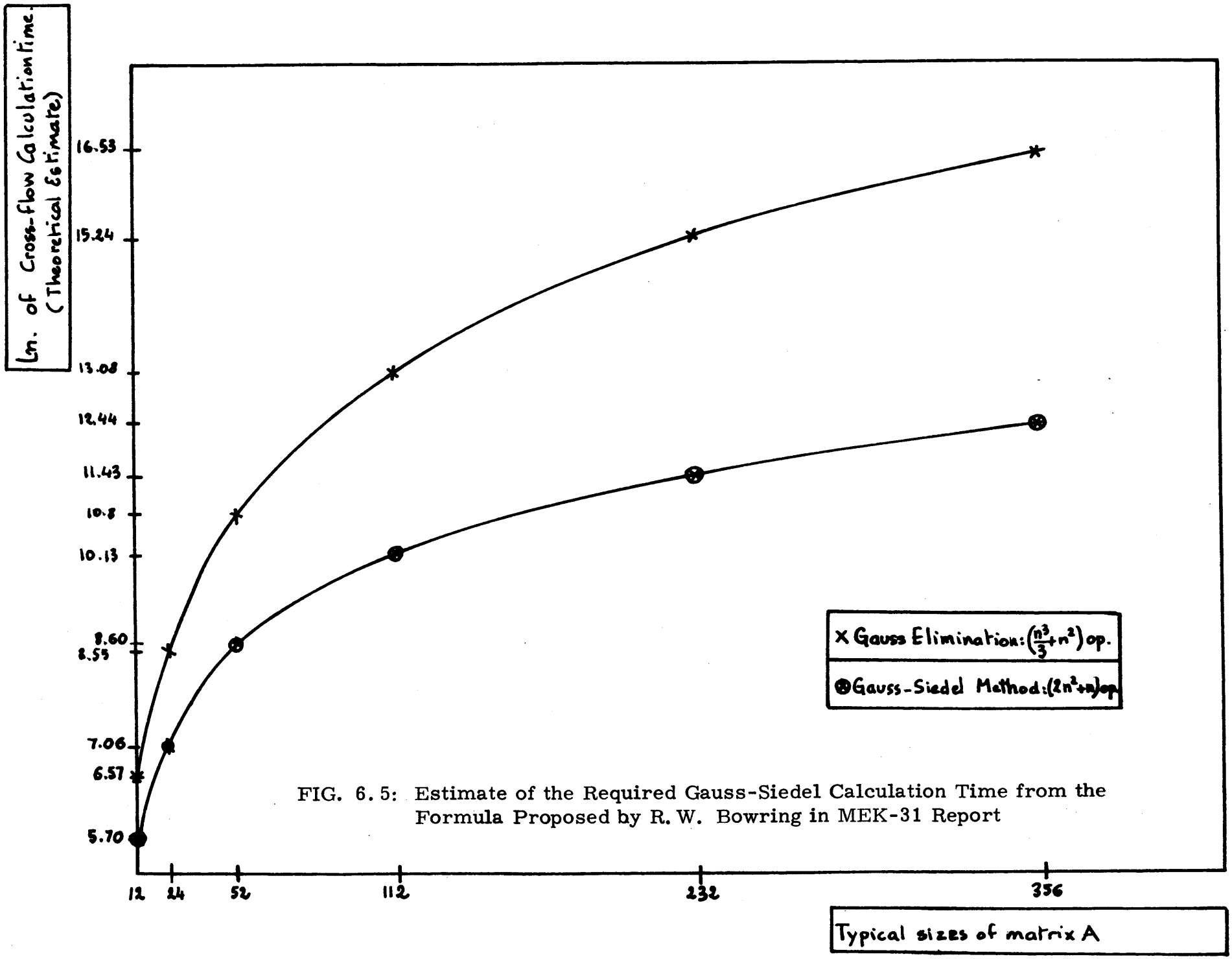


FIG. 6.5: Estimate of the Required Gauss-Siedel Calculation Time from the Formula Proposed by R. W. Bowring in MEK-31 Report

APPENDIX A

APPENDIX A

INTRODUCTION TO COBRA-IIIC: STUDY OF SUBROUTINE DIVERT

A.1 Channel Topography and Array LOCA

The objective of this section is to explain the stripping technique used to determine the coefficients of the cross-flows array AAA and to describe the information contained in the array LOCA. It is advised that one consult the report MEK-28[2] for additional information.

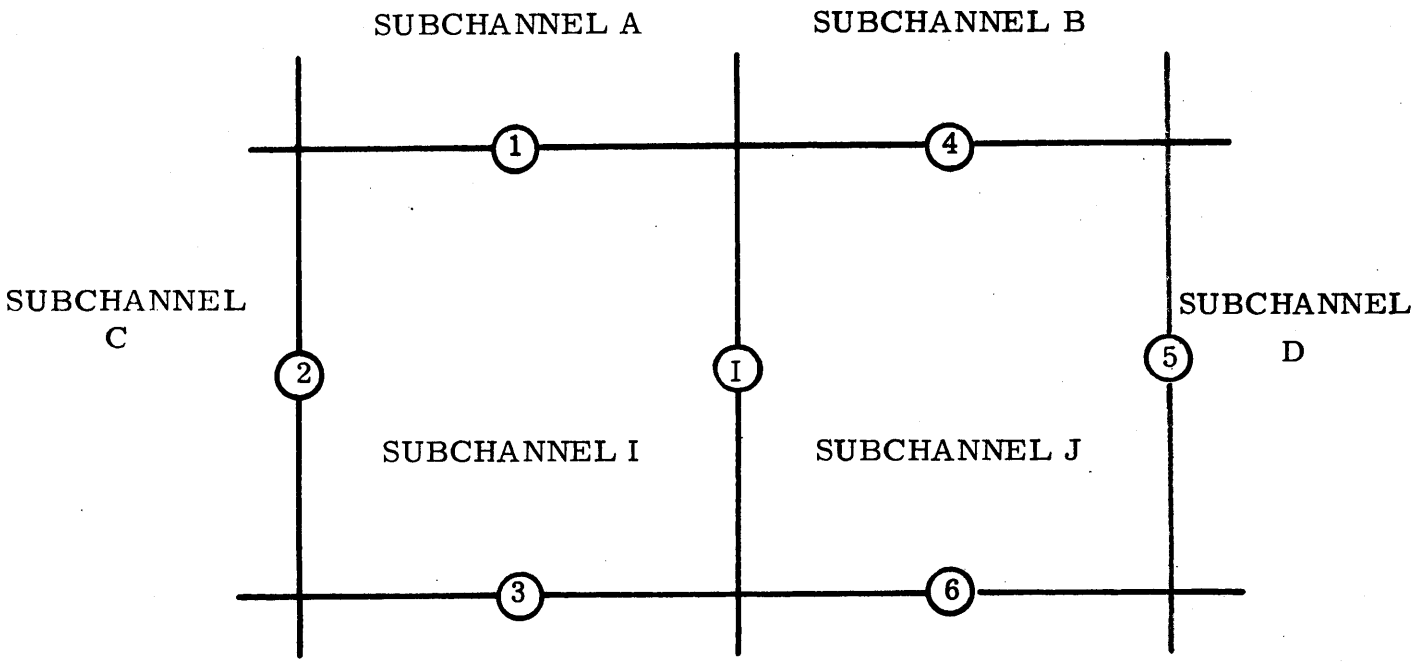
Assuming a square lattice geometry, as shown in Fig. A.1, the cross-flow w_{ij}, at boundary i-j, is affected by the cross-flows through the other boundaries of channels i and j. According to Fig. A.1 which corresponds to the most general location of two channels in the array, the cross-flow wij at the "principal boundary" i-j is influenced by the "secondary" boundaries: a-i, b-j, d-j, j-f, i-e, i-c. Therefore, all other cross-flow coefficients beyond the six secondary boundaries are set to 0.

It is noted that six is the maximum number of influencing secondary boundaries for a given boundary.

In the other two possible locations for a square array, i.e. edge and corner position, respectively, four and three are the number of influencing boundaries. (See Fig. A.2)

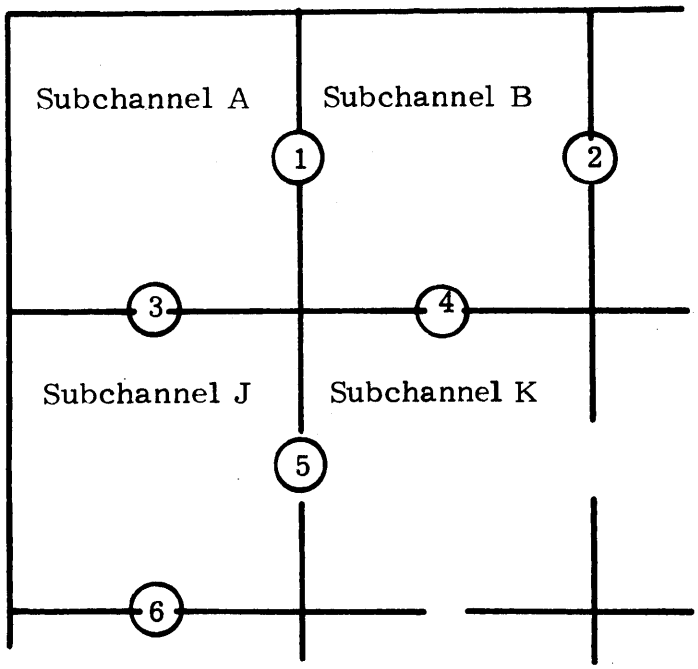
If NK represents the number of channel boundaries and if a_{ij} represents the coefficient of cross-flows for any i, j, $(1 < i, j < NK)$, a matrix, AAA, of cross-flows coefficients is formed.

The row position of the coefficients does not depend on the channel numbering but on the channel boundaries numbering.



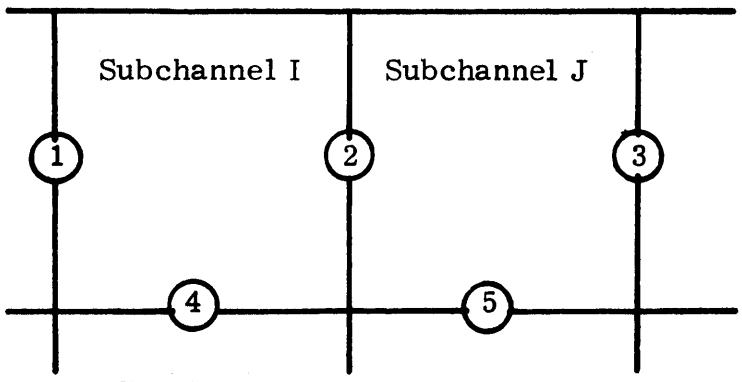
I	= boundary numbering of	I-J
1	same	A-I
2	same	C-I
3	same	I-E
4	same	B-J
5	same	J-F
6	same	J-D

FIG. A. 1 Subchannel Boundary Numbering



CORNER CASE: 3 Affecting Boundaries

1	affected by boundaries	2, 3, 4
3	affected by boundaries	1, 5, 6



EDGE CASE: 4 Affecting Boundaries

2	affected by boundaries	1, 3, 4, 5

FIG. A. 2: Corner and Edge Subchannel Case

It has been found that numbering the boundaries left to right, top to bottom for a pair of adjacent flow channels minimizes the width of the matrix interval in which the coefficients of the secondary boundaries are found. By this stripping technique the coefficients lie in a band along the diagonal having a width which is equal to twice the maximum of the difference between the diagonal and the extreme elements of a same row (i.e. the difference between the furthest element and the diagonal element) among all the NK rows of the matrix. Every element outside the striped band is, of course, equal to zero.

An additional argument for setting up the matrix AAA in a band-limited form can finally be presented: once the numbering scheme of the channel boundaries has been selected, the problem consists of placing in each row of the matrix AAA, whose diagonal element represents the considered cross-flow, all the other affecting cross-flows. An example in Fig. A.3 illustrates how to set up AAA.

Two sets of information are contained in the array LOCA:

i) sign of the cross-flows, and

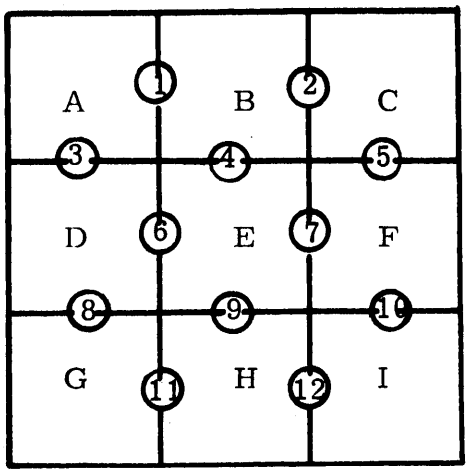
ii) numbering of the secondary affecting boundaries for each boundary in the set of given channels.

Discussion of Item i: The convention of sign presented in the MEK-28 report is summarized in Table A.1.

Discussion of Item ii: The array LOCA is a two dimension array (NK, 8) which provides the following information:

1) if the number of the principal boundary is K the corresponding FORTRAN statement becomes: LOCA (K, 1) = K

2) the number of all secondary boundaries is identified in the program by: (LOCA (K, L), L = 2, 7) i.e. secondary boundary, if none, the LOCA values are set to zero.



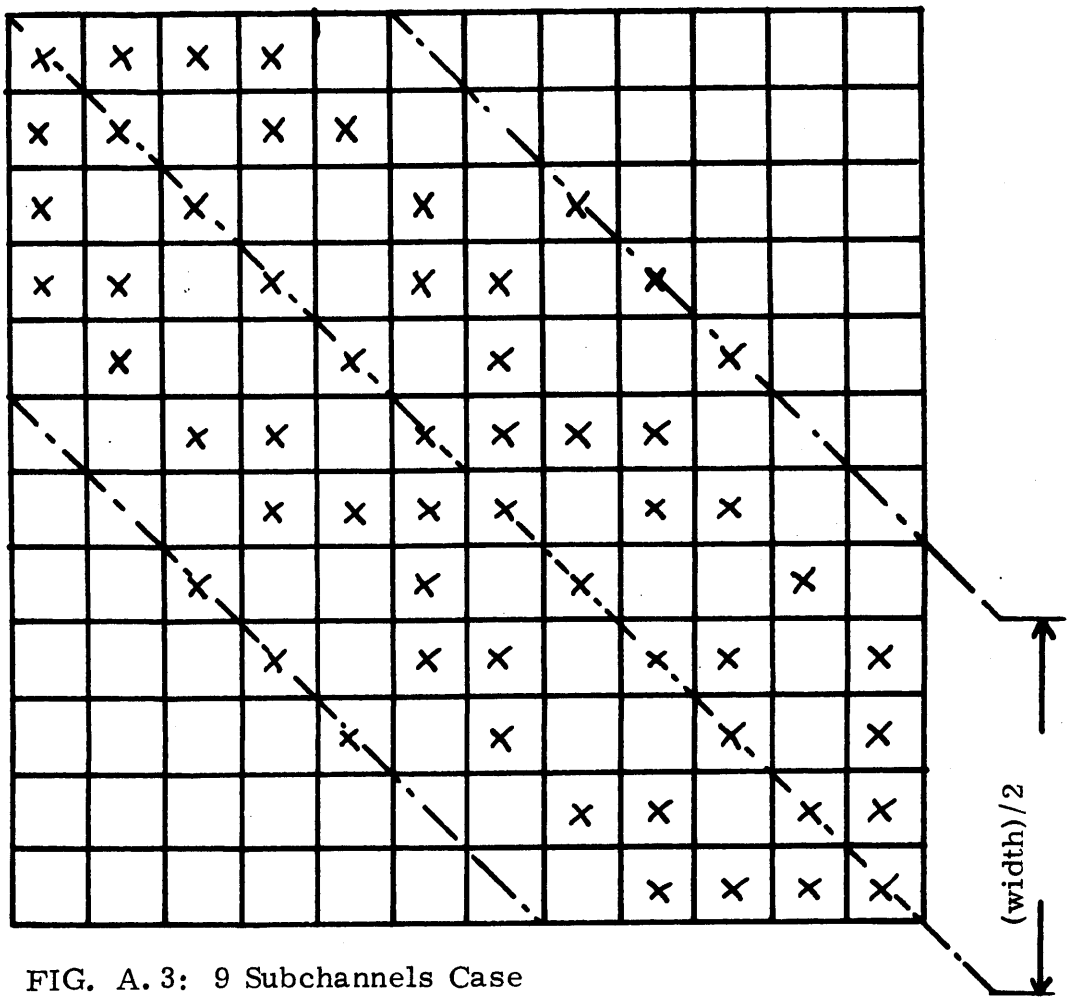


FIG. A.3: 9 Subchannels Case

TABLE A.1: Sign Convention used in DIVERT

1 - For Primary Boundary:

Sign of Cross-Flow	I J	I J
For Principal Boundary	0	0

2 - For Secondary Boundaries:

I - J, J - M	multiplied by
I J M or I J M	-1.0
otherwise	1.0

3) the total number of boundaries is written under the following FORTRAN statement: LOCA (K, 8)

For example, a 10 channel case, Fig. A.4 shows the boundaries numbering, and Table A.2 gives the numbering scheme used in LOCA. For example, the cross-flow at boundary 5 is affected by cross-flows through: 1, 2, 7, 8, 10. The array LOCA gives:

LOCA (5, 1) = 5
LOCA (5, 2) = -1
LOCA (5, 3) = 2
LOCA (5, 4) = 7
LOCA (5, 5) = -8
LOCA (5, 6) = -10
LOCA (5, 7) = 0
LOCA (5, 8) = 6

Finally, Table A.3 shows the information contained in LOCA for the 10-channel case.

A.2 Study of COBRA-III C Subroutine DIVERT

For greater clarity, the COBRA-III C coding and the subroutine DIVERT are referred to in this study respectively as COBRA and OND (Old-New-Divert).

The OND is composed of four parts:

i) lists of arrays and variables,
ii) setting the coefficients of the matrices AAA and B,
iii) solution of simultaneous equations by means of the Gauss Elimination, subroutine DECOMP and SOLVE,
iv) Modifying certain cross-flows if forced values.

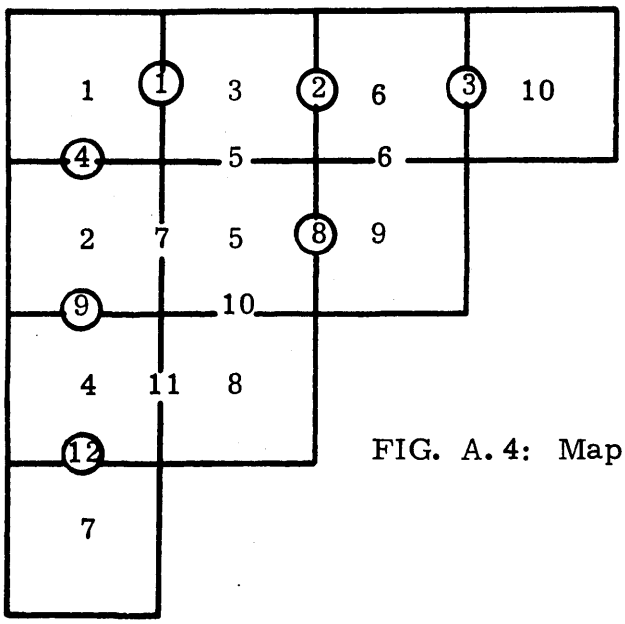


FIG. A. 4: Map for a 10 Channel Case

Table A. 2: Convention of Numbering used in LOCA

I	J	Boundary nber
1	3	1
3	6	2
6	10	3
1	2	4
3	5	5
6	9	6
2	5	7
5	9	8
2	4	9
5	8	10
4	8	11
4	7	12

LOCA(K,8) ARRAY SET IN ACOL K = 1 TO 12

(1)	1	2	3	4	5	6	7	8	9	10	11	12
(2)	4	-1	-2	1	-1	-2	-4	-5	-4	-5	-9	-9
(3)	-2	5	6	-7	2	3	9	-7	7	-7	12	11
(4)	-5	-3	0	-9	7	8	5	10	-11	8	10	0
(5)	0	-6	0	0	-8	0	-8	6	-12	11	0	0
(6)	0	0	0	0	-10	0	-10	0	0	0	0	0
(7)	0	0	0	0	0	0	0	0	0	0	0	0
(8)	4	5	3	4	6	4	6	5	5	5	4	3

MAXIMUM OVERALL STRIPE WIDTH FOR ABBEY AAA IN DIVERT = 11 FOR BOUNDARY NO. 10

REQUIRE 132 STORES FOR AAA SIZE AND THIS OK SINCE LESS THAN 1 PROVIDED

TABLE A.3: Output of the Array LOCA

Discussion of item i: Several variables and arrays used in the OND are listed under the following different names:

1) AAA, the matrix of cross-flow coefficients, has a elements ($1 \leqslant i, k \leqslant NK$) and these elements are represented by: DATA (SAAA + K + current index),
2) B, the NK-column matrix representing the "guessed" differential pressures, whose elements are b_{kj} ($1 \leqslant k \leqslant NK$, $1 \leqslant j \leqslant J$ = max number of axial steps) is listed as DATA (S B + index),
3) W, the NK-column matrix of the cross-flows whose elements are Wij, ($1 \leqslant i \leqslant NK$, $1 \leqslant j \leqslant J$ = max number of axial steps) is listed as DATA (S ANSWE + ind).

It is recommended that one read the MEK-20 [1] report for a complete list of variables and arrays used in COBRA.

Discussion of item ii: It is necessary to define the role of subroutine ACOL before detailing the second part of OND.

The maximum width of the band is determined in ACOL, and the value is assigned to MS. Then the subroutine CORE 2 (NK, MS), called by ACOL, reserves, as a dynamic storage with DATA instructions, (MS × NK) places in the core memory.

For example, in a 16-channel case, arranged in a square array, there are 24 channel boundaries and a maximum width of 11, then in the core memory 11 groups of 24 elements are reserved for the storage of AAA.

The objective of the second part of OND is to set up the coefficients of the AAA and B matrices and to arrange them in memory according to the following system of indices.

1. Every element in AAA is set up to zero by:
   DATA (SAAA + K+NK * (L-1) = 0
   with $1 \leq K \leq NK$, $1 \leq L \leq MS$.
2. Setting elements of B (B is an NK-column matrix) after having calculated them is performed by the statement:
   DATA (SB + K) = f (variables depending on pressure),
   with $1 \leq K \leq NK$.
3. Setting the elements of AAA outside the principal diagonal, using LOCA (K, 8), for ($1 \leq K \leq NK$), which stores the maximum number of secondary affecting boundaries for boundary K, (see Fig. A. 5), is performed according to the instructions
   NBOUND = IDAT (SLOCA + K + MG*7),
   with MG = maximum number of boundaries.
   The variable LL is an index varying between 1 and NBOUND and allows then the current index to be computed by
   L = IDAT (SLOCA + K + MG * (LL-1).
   L only varies thereby up to the last significant value in the array LOCA. Since (LL-1) can have value between 0 and (NBOUND-1), a test on LL is made in order to protect the first significant coefficient in the (MS NK) array for the row K.
   DATA (SAAA + K + NK * (L-1) , where in this case
   L = MID-K+L and MID = (MS+1)/2.
   Note that the sign of the coefficient is restored by SAVE.
   Now, for all other values of LL the coefficients are stored in the array positions
   DATA (SAAA + K + NK * (L-1) , $1 \leq K \leq NK$.

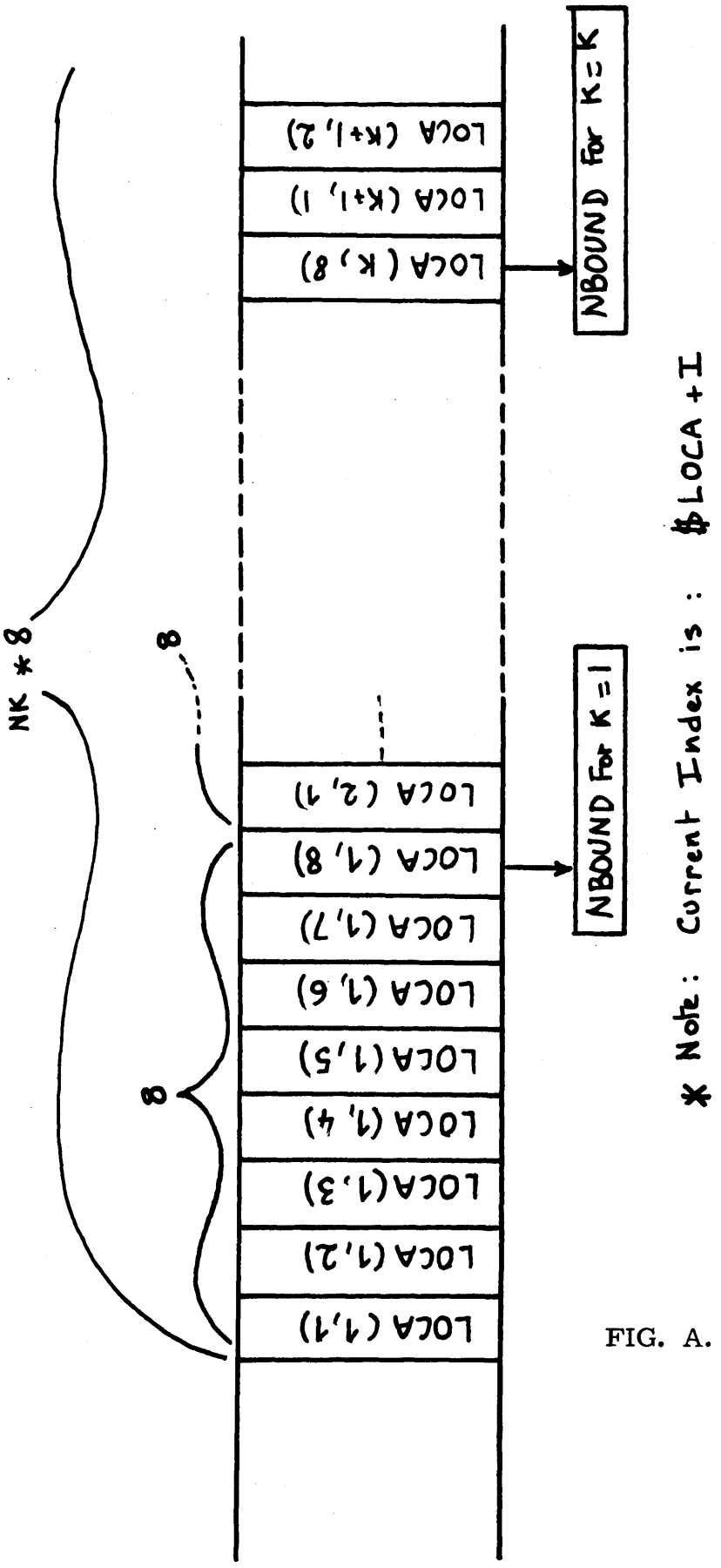


FIG. A. 5: Storage of the array LOCA

4. The elements of the principal diagonal in the original array (NK×NK) are stored in the column MID of the array (MS × NK), with MID = (MS + 1)/2.

   Matrix elements are then stored by:
   DATA (SAAA + K + NK ✱ (MID-1), $1 \leq K \leq NK$.
   Note that the array (NK × NK) has always been "virtual", and never used nor indiced under this ideal form.
   The rearrangement of the elements in (MS×NK) storage, compared to the initial location they would have had in (NK × NK) is shown in Tables A. 4 and A. 5.

   One can conclude that the motivation in storing the nonzero elements of the AAA matrix under an (MS×NK) array is to relocate the significant coefficients of the cross-flows "around" the element MID of the row, according to the information given by LOCA. However, it should be noticed that this configuration does not avoid an important number of element 0 per row.

Discussion of item iii: This part of OND deals with the modifications of simultaneous equations to account for specified values of cross-flows given in subroutine FORCE. Since the different case of channels set did not involve any forced cross-flows this part has not been changed.

Discussion of item iv: Subroutine DECOMP is first called by OND and uses the method of maximum pivoting for triangularing the array (NK×MS). By successive transformation, this array is put into a form shown in Fig. A. 6.

It should be noted that the pivoting is made around the MID column of the previous array. A test for the singularity of the system is made for the element MID of the last row NK. If this is 0, the matrix is then singular and the system cannot be solved.

TABLE A.4: Central Part of Array (NK x NK)

-2.8564E-02	1.3CC0E-01	2.3437E-03	-2.1700E-01	-1.0296E-01		
RRAY AAA:						
0.0	0.0	C.C	C.C	0.C	1.3234E+01	-6.4059E+00
C.0	6.4054E+00	-6.4C59E+00	0.0			
C.C	C.0	C.C	0.0	-6.4060E+00	1.3235E+01	-6.4061E+00
0.0	6.4059E+C0	-6.4C59E+C0	0.C			
C.C	0.0	0.0	0.0	-6.4059E+C0	1.9636E+01	0.0
C.C	6.4C59E+C0	C.C	0.0			
0.0	0.0	6.4C52E+C0	C.C	0.0	1.3233E+01	0.0
C.C	-6.4042E+00	0.0	-6.4045E+C0			
0.C	-6.4C60E+00	6.4059E+00	0.0	0.0	1.3235E+01	0.0
6.4064E+00	-6.4062E+00	C.C	-6.4C63E+C0			
C.C	-6.4059E+00	6.4061E+C0	0.0	0.C	1.9642E+01	C.0
1.2813E+01	C.0	C.0	0.0			
0.0	0.0	-6.4C42E+C0	6.4C63E+C0	0.0	1.3234E+01	-6.4062E+00
6.4045E+00	-6.4063E+00	C.0	0.0			
0.C	C.0	-6.4063E+00	1.2814E+01	-6.4064E+00	1.9643E+01	0.0
6.4C63E+00	0.0	C.C	0.0			
-6.4042E+00	0.0	0.0	6.4042E+C0	0.C	1.3229E+01	0.0
-6.4C17E+C0	-6.4019E+C0	C.0	0.0			
-6.4063E+00	0.0	-6.4064E+C0	6.4062E+C0	0.0	1.9642E+01	1.2813E+01
0.0	0.0	C.C	0.C			
C.C	0.0	C.0	-6.4019E+00	1.2812E+01	1.9638E+01	6.4019E+00
0.C	0.0	C.C	0.0			
C.0	0.0	-6.4C19E+C0	0.0	6.4017E+C0	1.9632E+01	0.0
C.C	C.0	0.0	0.0			

TABLE A. 5: Output of the Array (NK x MS)

-2.8564E-02 1.5CC0E-01 2.3437E-03 -2.7700E-01 -1.6258E-01
AY AAA:

0.0	0.0	C.C	C.C	0.C	1.3234E+01	-6.4059E+00
C.C	6.4054E+00	-6.4059E+00	0.0			
C.C	0.0	C.C	0.0	-6.4060E+00	1.3235E+01	-6.4061E+00
0.C	6.4059E+C0	-6.4059E+C0	0.C			
C.C	0.0	0.0	0.0	-6.4059E+C0	1.9636E+C1	C.0
C.C	6.4C59E+CC	C.C	0.0			
0.0	0.0	6.4C52E+C0	C.C	0.0	1.3233E+01	0.0
C.C	-6.4042E+00	0.0	-6.4045E+C0			
C.C	-6.4C60E+00	6.4059E+00	0.0	0.0	1.3235E+01	0.0
6.4064E+00	-6.4C62E+00	C.C	-6.4C63E+C0			
C.C	-6.4059E+00	6.4061E+C0	0.0	0.C	1.9642E+01	C.0
1.2813E+01	C.0	C.0	0.0			
0.0	0.0	-6.4C42E+C0	6.4C63E+C0	0.0	1.3234E+01	-6.4062E+C0
6.4045E+00	-6.4063E+00	C.0	0.0			
0.C	C.0	-6.4C63E+00	1.2814E+C1	-6.4064E+00	1.9643E+01	0.0
6.4C63E+C0	C.0	C.C	0.0			
-6.4042E+C0	0.0	0.C	6.4042E+C0	C.C	1.3229E+01	0.0
-6.4C17E+C0	-6.4C19E+C0	C.0	0.0			
-6.4063E+C0	0.0	-6.4C64E+C0	6.4062E+C0	0.0	1.9642E+01	1.2813E+01
0.0	0.0	C.C	C.C			
C.C	0.0	C.0	-6.4019E+C0	1.2812E+01	1.9638E+01	6.4019E+00
C.C	C.0	C.C	0.0			
C.0	0.0	-6.4C19E+C0	0.C	6.4017E+C0	1.9632E+01	0.0
C.C	C.0	0.0	0.0			

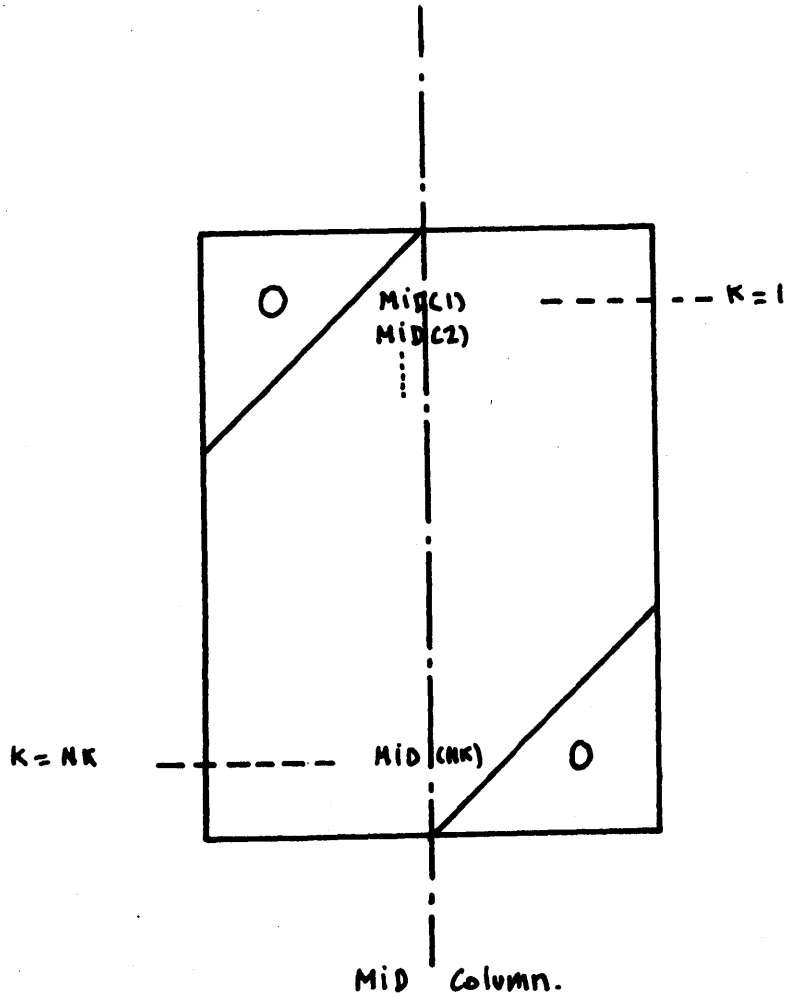


Fig. A.6: Transformation of the array (MS * NK) by Subroutine DECOMP.

Subroutine SOLVE is then called for the resolution of the linear system. Subroutines DECOMP and SOLVE are two complementary subroutines of the Gauss Elimination method used to solve the matrix Eq. 1.1.

Then, the results, for the given core axial step J,are stored for print outs in the array.

DATA (SW + K + MG * (J-1) = DATA (SANSWE + K),

where

DATA (SANSWE + K) is the cross-flow solution at row K, for the axial step J.

APPENDIX B

APPENDIX B

I Background

One recalls that the linear relation between the cross-flows of a same axial step J can be written down

$$A \ \underline{X}_J = \underline{B}_J \qquad (1.1)$$

where

A is the cross-flow coefficient matrix at J,

$\underline{X}_J$ is the cross-flow "vector" at axial step J, and

$\underline{B}_J$ is the pressure differential between axial step J and J-1.

In the most general location of a set of two subchannels in the array, a particular cross-flow may be affected by a maximum of seven other cross-flows in its immediate vicinity. The other cross-flows at secondary boundaries have no effect on the particular cross-flow and are then set to 0.

Therefore, the relation (1-1) which can be written

$$\sum_{\ell} a^j_{i\ell} \ x^j_{i\ell} = b^j_i \qquad \begin{matrix}(i = 1,\text{--}\,NK)\\(j = 1,\text{--}\,J)\end{matrix}$$

can be interpreted as involving only a maximum of seven nonzero coefficients in each row of A. It should be also noted that the configuration of A, i.e. the place of the nonzero elements in each row, is directly related to the chosen boundary numbering scheme in the problem. It has been found that numbering the boundaries from left to right and from top to bottom decreases for each row the width of the interval within which the significant coefficients are found. With

this procedure one ends up with a band matrix.(2)

Note also that by symmetry of construction, the matrix A is symmetric.

It has been noticed that the Gaussian Elimination method used to solve the cross-flow problem in the subroutine DIVERT of COBRA-III C resulted in large computational time. Moreover, if large order matrices would be considered, this could be a major concern in the computational budget.

Because of the sparsity of A, it has been thought that an iterative procedure particularly adapted to this type of problem should be investigated.

This report presents the successive over relaxation method (SOR) developed for this particular type of matrix, and its corresponding coding to be included in the MIT-modified version of the COBRA III-C code. With this new method, another major modification has been brought to subroutine DIVERT: as noted before, a maximum of seven nonzero elements lies among each row of MS-elements of the band matrix. The storage of the band width matrix is done by the subroutine CORE 2 (NK, MS) in the COBRA III-C version and reserves (NK × MS) spaces in the core memory. Obviously, this storage is oversized since only a maximum of (7 × NK) spaces will contain nonzero elements. For large order matrices, one then notes, the saving in storage would be significant if only (7 × NK) spaces were affected to the storage of A. The generation of the array A under a compact form constitutes the second major modification to the code and the corresponding program is detailed in this report.

Finally, for programmers unfamiliar with COBRA, a list of array, indices & variables used throughout the different modified subroutines is also given.

II Successive over Relaxation Method (SOR)

This section is intended to provide the basic equations used for the set-up of the problem. For additional references, one may consult one of the sources listed in **References.**

At each axial step, i. e. each axial plane containing a cross section of the subchannels, one has the following relation between the cross-flow vector $\underline{W}_J$, the pressure differential vector $\underline{P}_J$, and the cross-flow coefficient matrix AAA :

$$AAA_J \quad \underline{X}_J = \underline{P}_J \qquad \text{for } J = 1, \text{----} N. \tag{1}$$

Splitting the coefficient matrix AAA_J, such as:

$$AAA_J = -E + D - F \tag{2}$$

the Gauss-Siedel procedure gives for the kth iteration:

$$D \ \underline{W}_J^{(k)} = \underline{P}_J + E \ \underline{W}_J^{(k)} + F \ \underline{W}_J^{(k-1)} \tag{3}$$

Since the coefficient $a_{ii} \neq 0$, for all i = 1, ---- NK, this insures that (D - E) is non-singular and (3) becomes:

$$\underline{W}_J^{(k)} = (D - E)^{-1} F \ \underline{W}_J^{(k-1)} + (D - E)^{-1} \underline{P}_J \tag{4}$$

k = 0, 1, 2,

Now, if one wishes to hasten the convergence of the solution, one may consider the successive over relaxation technique. The new value of $W^{(k)}$ is taken to be:

$$\underline{W}_J^{(k)} = \underline{W}_J^{(k-1)} (1 - \Omega) + \underline{W}_J^{(k)} \tag{5}$$

for some parameter Ω such as $1 < \Omega < 2$, for which the number of iterations becomes minimum.

It may prove useful to formulate the problem in terms of component $W_{i,j}$ of the vector $\underline{W}_J$.

1) The Gauss-Siedel method at the (k+1)th iteration, gives:

(next page)

$$W_{i,J}^{(k+1)} = \frac{1}{a_{ii,j}} \left(p_{i,j} - \sum_{\ell=1}^{i-1} a_{i\ell,J} W_{\ell,J}^{(k+1)} - \sum_{\ell=i+1}^{NK} a_{i\ell,J} W_{\ell,J}^{(k)} \right)$$

$$\text{for } J = 1, \text{----} N$$

$$i = 1, \text{----} NK \qquad (6)$$

and

2) the SOR method gives:

$$\tilde{W}_{i,J}^{(k+1)} = (1 - \Omega) W_{i,J}^{(k)} + \Omega W_{i,J}^{(k+1)} \quad . \qquad (7)$$

The coding of the iterative technique for a cross-flow solution is based on the Eq. 7 and is detailed in the following pages.

B. 1 List of Arrays, Variables, Indices used in the Subroutine for Iterative Method, ITER

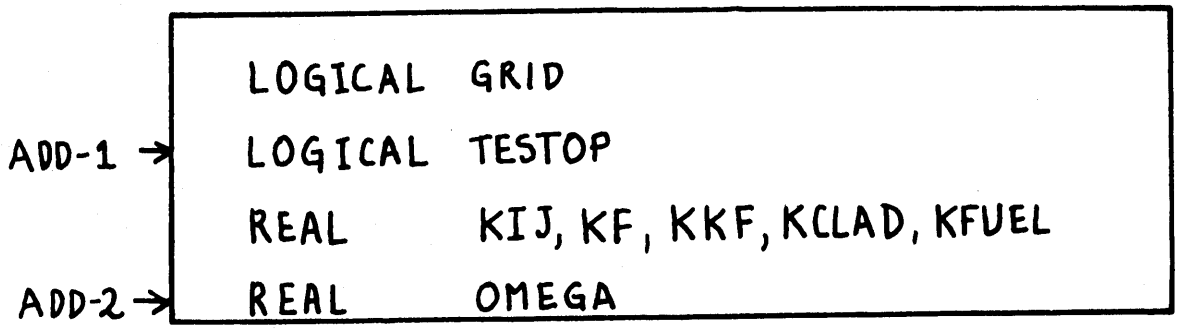
Note that the code name of the iterative procedure is ITER. For a better understanding of the extent of the transformations brought to DIVERT to include an iterative procedure it has seemed useful to present in this report a complete list of arrays and variables even if most of them do not differ from those used in DIVERT.

Note also that the modifications are labeled by: ADD i. e. variables added to the list.

```
      IMPLICIT INTEGER (A-Z)
      COMMON / COBRA1 / ABETA , AFLUX , ATOTAL , BBETA , DIA , DT , DX ,
     1    ELEV , FERROR , FLO , FTM , GC , GK , GRID , HSURF , HF ,
     2    HFG , HG , I2 , I3 , IERROR , IQP3 , ITERAT , J1 , J2 ,
     3    J3 , J4 , J5 , J6 , J7 , KDEBUG , KF , KIJ ,
     4    NAFACT , NARAMP , NAX , NAXL , NBBC , NCHAN , NCHF , NDX , NF ,
     5    NGAPS , NGRID , NGRIDT , NGTYPE , NGXL , NK , NODES , NODESF , NPROP ,
     6    NRAMP , NROD , NSCBC , NV , NVISCW , PI , PITCH , POWER , PREF ,
     7    QAX , RHOF , RHOG , SIGMA , SL , TF , TFLUID , THETA , THICK ,
     8    UF , VF , VFG , VG , Z ,
      COMMON / COBRA2 / AA(4) , AF(7) , AFACT(10,10) , AV(7) , AXIAL(30) ,
     1 AXL(10) , BB(4) , BX(30) , CC(4) , CCLAD(2) , CFUEL(2) , DFUEL(2) ,
     2 GAPXL(10) , GFACT(9,10) , GRIDXL(10) , HGAP(2) , HHF(30) , HHG(30) ,
     3 IGRID(10) , KCLAD(2) , KFUEL(2) , KKF(30) , NCH(10) , NGAP(9) ,
     4 PP(30) , RCLAD(2) , RFUEL(2) , SSIGMA(30) , TCLAD(2) , UUF(30) ,
     5 VVF(30) , WG(30) , XQUAL(30) , Y(30) , TT(30)
```

Note that the two COMMON lists above have not been modified. They are identical to those used in DIVERT of the COBRA-III C version.

Note also that FERROR and FLO are used later as "weighting" parameters.



TESTOP: ADD-1: TESTOP has a logical value. It can be . TRUE. or . FALSE.. This is used, after the comparison test to a selected criterion, to determine whether the test is effectively satisfied or not for all cross-flows at a certain axial step J. Note that TESTOP is initialed to . TRUE.. The way the iterative method is programmed implies that if one and only one cross-flow does not reach, at any iteration, the desired result, i. e. if it fails to satisfy the convergence criterion, the logical TESTOP will then have the value . FALSE., and the iterative procedure will be repeated until satisfactory results are reached for each of the cross-flows.

OMEGA: ADD-2: REAL OMEGA; OMEGA defined as a real number is the computed value of the parameter of the successive over relaxation procedure. This has been found by increasing, between 1 and 2, a number by 0.01 at each

step, in order to find the value which minimized the number of iterations for the successive over relaxation formula: 4/3 is the value of OMEGA which minimizes iterations and then computation time.

```
      COMMON/COBRA3/ MA , MC , MG , MN , MR , MS , MX,
     1          $$$ , $A , $AAA, $AC , $ALPHA , $AN , $ANSWE , $B,
     1    $CCHAN , $CD , $CHFR , $CON , $COND , $CP , $D, $DC , $DFDX,
     2    $DHDX, $DHYD , $DHYDN, $DIST , $DPDX, $DPK , $DUR , $DR, $F,
     3    $FACTO, $FDIV , $FINLE , $FLUX, $FMULT, $FOLD, $FSP, $FSPLI, $FXFLO
     4    $GAP , $GAPN , $GAPS , $H , $HFILM , $HINLE, $HOLD, $HPERI, $IDAR
     5    $IDFUE , $IDGAP , $IK , $JBOIL , $JK , $LC , $LENGT, $LOCA, $LR,
     6   $NCHFR, $MCFAC , $MCFRR , $NTYPE , $NWRAP, $NWRPS, $P, $PERIM, $PH
     7   $PHI, $PRNTC, $PRNTR, $PRNTN, $PW, $PWRF, $QC, $QF, $QPRIM,
     8   $QUAL, $RADIA, $RHO , $RHOOL, $SP , $T , $TDUMY, $TINLE, $TROD,
     9   $U , $UH , $USAVE , $USTAR , $V , $VISC , $VISCW, $VP, $VPA ,
     A   $W, $WOLD, $WP , $WSAVE , $X , $XCROS
      COMMON DATA(1)
      LOGICAL LDAT(1)
      INTEGER IDAT(1)
      EQUIVALENCE (DATA(1), IDAT(1), LDAT(1))
      EQUIVALENCE (NCHAN, NCHANL)
```

Note that, from the above list, no variables have been modified, added or withdrawn.

Note: NCHANL is used in the convergency criteria.

The new variables defined for ITER are discussed below:

EPS

EPS is the value assigned to the convergency criterion. It has seemed useful to present in this report the two convergency criteria which may be considered.

1 - Relative Criteria

For the (p+1) th iteration, one ends up with:

$$\frac{\left| W^{(p+1)} - W^{(p)} \right|}{\left\| W^{(p)} \right\|} \leq EPS$$

It is the adequate convergency criterion for the Gauss-Siedel method.

According to the accuracy of the results in which one may be interested the value assigned to EPS can be "weighted" by outside parameters such as the axial flow in the channel, an average of the flows in two adjacent channels, or even by small real numbers.

However, after different runs, the "nominal" value of EPS has been taken to be 10^{-3}, which permits both sufficient accuracy of the results, and reasonably small calculational times.

Note that for this relative convergency, one has to be sure of the positive value of the denominator. In fact, after a certain number of iterations, one can end up with a very small

value of some of the cross-flow components, i.e. $|W^p| \simeq 10^{-20}$. The effect on the calculational time is clear: one will have very large computational requirements and therefore costly runs.

The denominator has to be "proofed" against such cases: it suffices to add to the value $|W^p|$, a very small positive number.

When used in this subroutine, the test was:

II. Absolute Criteria

One can also use the absolute difference between two successive iterations as a convergence criterion, i.e.:

$$|W^{p+1} - W^p| < EPS$$

This test is easier to handle from a computational viewpoint, since it may lead to shorter calculation time but one must consider that it is not as strict as the relative criteria: in case of non-uniform convergence, the iterations may be terminated further from the exact result than those obtained with the relative criteria.

As in case 1, different outer parameters have been tried, if one wants greater accuracy in the results. As before, the nominal value of EPS is fixed to be 10^{-3}:

1- EPS x FERROR = EPS. Since FERROR, which is the allowed error for flow calculation

is of the order of 10^{-3}, the test is carried out with a precision to 10^{-6} between two successive iterations. If the results have improved in accuracy, the calculation requirement has been too large, and this has been cancelled out.

2- EPS ✕(FLO/NCHANL) has been tried in order to get a test value compatible with the order of magnitude of the flow in the channel. Also, one ends up with unrealistically large calculation times.

3 - An average value of the flow in two adjacent channels has also been tried as a convergence criterion for the same reason, but once again, the calculation time ended up to be large.

IITER — This variable represents the number of iterations. The maximum allowed value is 200. If this value is reached, the calculation is stopped - for the particular cross-flow at a given axial step - and the results are printed out.

Note that this number is intended to protect the calculation time, in case of an error in the input.

INDI — represents the value of the index of the nonzero coefficient in the array AAA and the index of the corresponding value of the cross-flow to be multiplied to this coefficient.

TERMM has the value for each iteration of:

$$\sum_{\ell-1}^{i-\ell} a_{ie,j} W_{e,j}^{(k+1)} - \sum_{\ell+1}^{NK} a_{ie,j} W_{e,j}^{(k)}$$

This is computed through the index INDI.

NTR is an index such as $2 < NTR < 7$ and is intended to find, through LOCA, the values of the coefficients and cross-flows, forming the expression (7).

RESERV is the value attributed to the previous value of the component W_i, at iteration k, if the running iteration is at (k + 1).

RAPID becomes the absolute difference between

$$\left| W_e^{(k+1)} - W_e^{(k)} \right|$$

or the relative absolute difference

$$\left| \frac{W_e^{(k+1)} - W_e^{(k)}}{W_e^{(k)}} \right|$$

according to the chosen convergency criteria.

B. 2 Generation of AAA

As mentioned before, the space attributed in the core memory by DIVERT to the storage of AAA, is by far too large. It is useful to look back to this previous set-up of the cross-flow coefficients array in order to understand the present modifications.

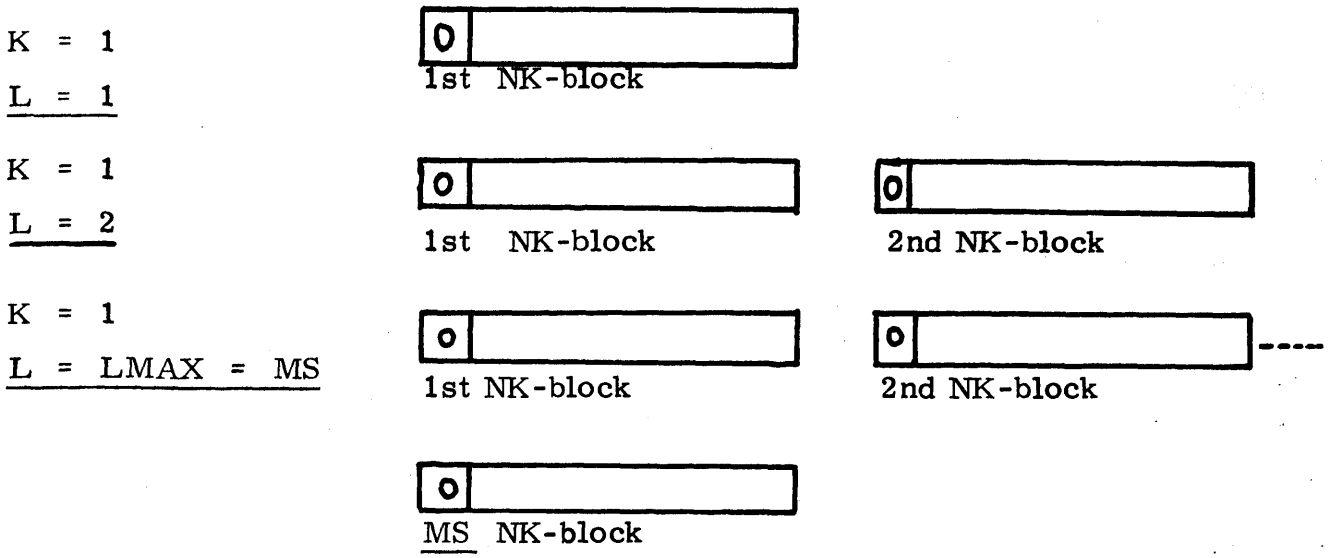
It consisted of:

1. Set-up (NK ✕ MS) places in the core memory to 0 by:

```
      LMAX = MS
      DO 310 K = I, NK
      DO 290 L = I, LMAX
290 DATA (  AAA + K + NK * (L-1)) = 0
310 CONTINUE
```

Note that LMAX, or MS, represents the computed value of the maximum width of the band striped matrix. This is done in the subroutine ACOL, through CORE 2 (NK, MS).

Note also that the way of storing AAA is done in the following way: MS blocks of NK elements are set up to 0; with the same location, in an NK-block, being set to 0 at each stage of the outer DO LOOP.



2. Set-up of vector B

Note that this part has not been modified.

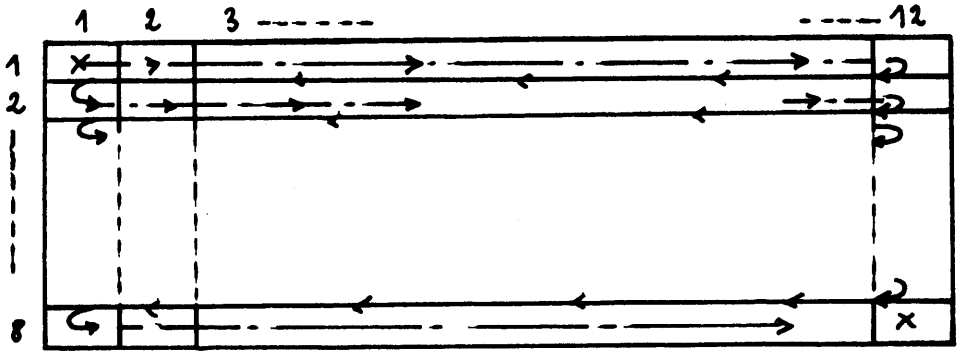
```
      II = IDAT ($IK+K)
      JJ = IDAT ($JK+K)
      DATA ($B+K) = (DATA ($SP+K+MG*(J-1))-(DATA ($DPDX+II)-DATA ($DPDX+JJ)*
     1      DX)*SL*DATA ($FACTO+K)+DATA ($USAVE+K)*DATA ($W+K+MG*(JM1-1))/
     2      DXGC+DATA ($WOLD+K+MG*(J-1))/DTGC.
      SAVE = ABIT(1, DATA ($U+II), DATA ($USTAR+K), DATA($A+II), DATA($DPK+II),
     1      DATA ($F+II+MC*(JM1-1), DATA ($F+II+MC*(J-1)))
     2 +ABIT(1, DATA ($U+JJ), DATA ($USTAR+K), DATA ($A+JJ), DATA ($DPK+JJ),
     3      DATA ($F+JJ+MC*(JM1-1), DATA ($F+JJ+MC*(J-1)))
```

3. Find the maximum number of affecting cross-flow for each cross-flow i ($1 \leqslant i \leqslant NK$) by:

```
NBOUND = IDAT (    LOCA + K + MG * 7)
```

It has seemed useful to detail the mechanism of this instruction. LOCA is an (NK × 8) or an (MG × 8) array. The 8 th element in each column, is the maximum number of affecting cross-flow for a given one. Since an array is always stored by row, the twelve last elements will be found by:

MG * 7 + K with $1 \leqslant K \leqslant NK$.



4. Once NBOUND (K) is found for K, an index LL will vary between 1 and NBOUND (K) to find the value of the affecting cross-flow by:

```
DO 300 LL = 1, NBOUND
```

TABLE B.1: Output of the Array LOCA

LOCA(K,8) ARRAY SET IN ACOL K = 1 TO 12

(1)	1	2	3	4	5	6	7	8	9	10	11	12
(2)	4	-1	-2	1	-1	-2	-4	-5	-4	-5	-9	-9
(3)	-2	5	6	-7	2	3	9	-7	7	-7	12	11
(4)	-5	-3	0	-9	7	8	5	10	-11	8	10	0
(5)	0	-6	0	0	-8	0	-8	6	-12	11	0	0
(6)	0	0	0	0	-10	0	-10	0	0	0	0	0
(7)	0	0	0	0	0	0	0	0	0	0	0	0
(8)	4	5	3	4	6	4	6	5	5	5	4	3

MAXIMUM OVERALL STRIPE WIDTH FOR ABBAY AAA IN DIVERT = 11 FOR BOUNDARY NO. 10

REQUIRE 132 STORES FOR AAA SIZE AND THIS OK SINCE LESS THAN 1 PROVIDED

Note that NBOUND is of course different for different K.

L = IDAT (LOCA + K + MG (LL-1))

Now L represents the value of the element found in the array LOCA. If, for example, in the case of 10 channels, 12 boundaries (see fig. 2), one takes the 7th column of NK.

LOCA (7, 1) = 7

LOCA (7, 2) = -4

⋮

Note that, one always has: LOCA (K, 1) = K.

For the first value of LL (LL=1), the cross-flow coefficient value is on the principal diagonal and this is done in two steps.

STEP 1: L = MID - K+L

Note that for any K, ($1 \leqslant K \leqslant NK$), L in the RHS will always have the value K. Then L (LHS) is always equal to MID. This is done on purpose since, in this subroutine, one wants to store all the diagonal elements of the matrix AAA, i.e. the element of the NK column MID, in the same NK-block by the instruction:

DATA (AAA + K + NK * (L-1)) = SAVE * -----

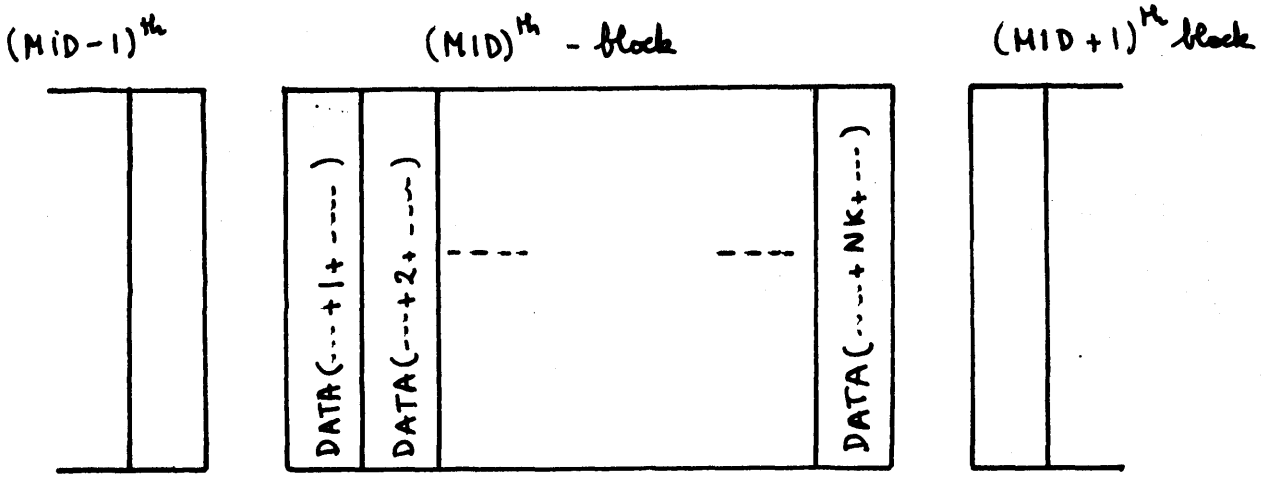
Note that this instruction is valid also for any other values of K, since it affects their value. Only at the end of the DO Loop, on the affecting cross-flow (LL), the diagonal element is corrected by:

DATA (AAA + K + NK * (MID - 1)) = DATA (AAA + K + NK * (MID - 1) + ------ R

R in RHS is nothing else but:

DATA (AAA + K + NK * (L - 1)) for L = MID

It is interesting to note that the MID elements are stored in a same NK-block:



STEP 2:

For the other elements corresponding to LL = 1, an index L, having the value of the corresponding element in LOCA is given by:

L = IDAT (LOCA + K + MG * (LL-1)) .

The value of L can then be negative or positive. An absolute value is taken:

L = IABS (L)

and the value of the index becomes

L = MID - K + L

Since for this particular case MS = 11, NK = 12, MID is found to be 6 (from ACOL), one gets L in (LHS).

Therefore, (MS - 1) blocks of NK-elements are used for the storage of a maximum of (7 x NK) nonzero elements.

One can then notice that the place in the core memory is oversized since MS can be found very large for the same amount, i. e. 7 x NK nonzero elements in the array.

Setting a New Compact Array (7 NK)

The idea is to reduce the storage by relocating the elements of the matrix AAA according to a bijection with the elements of LOCA.

In ITER the adopted procedure is the following:

STEP 1:

Set-up a zone of (7 X NK) elements in the core memory to 0 by:

```
      DO 1975 K = 1, NK
      DO 2222 L = 1,7
2222 DATA ( AAA + K + NK * (L - 1)) = 0
```

STEP 2:

Find the maximum number of affecting cross-flow in LOCA by:

NBOUND = IDAT (LOCA + K + MG * 7)

DO 1974 LL = 1, NBOUND

L = IDAT (LOCA + K + MG * (LL - 1)

same as DIVERT

L = IABS (L)

same as DIVERT

STEP 3:

The main difference is to give to the value of the cross-flow coefficients a location in the (7 X NK) places, by:

```
1976        DATA ( AAA + K + NK * (LL - 1)) = - - - -
1974        CONTINUE
```

Note that, now there is no difference between the different value of LL - i.e. if LL is a diagonal element or any element of a row - because MID - the NK-column - around which the location of each element of a row was made, is no longer used. The inner DO LOOP - on 1974 - is done on a particular column of LOCA and gives by LL the direct location that must be affected to the corresponding element of the array AAA. Note also that thereafter if it is an element of the principal diagonal its value is

corrected by:

DATA (AAA + K) =

1 DATA (AAA + K) + -----

2

This corresponds to the case LL = 1, i.e. LL - 1 = 0.

Remember that one always has LOCA (K, 1)= K

Then, by

1975 CONTINUE

THE outer DO Loop on the K (1≤ K≤ NK) is pursued.

Note that with this relocation of the elements of AAA one ends up with Table B. 3.

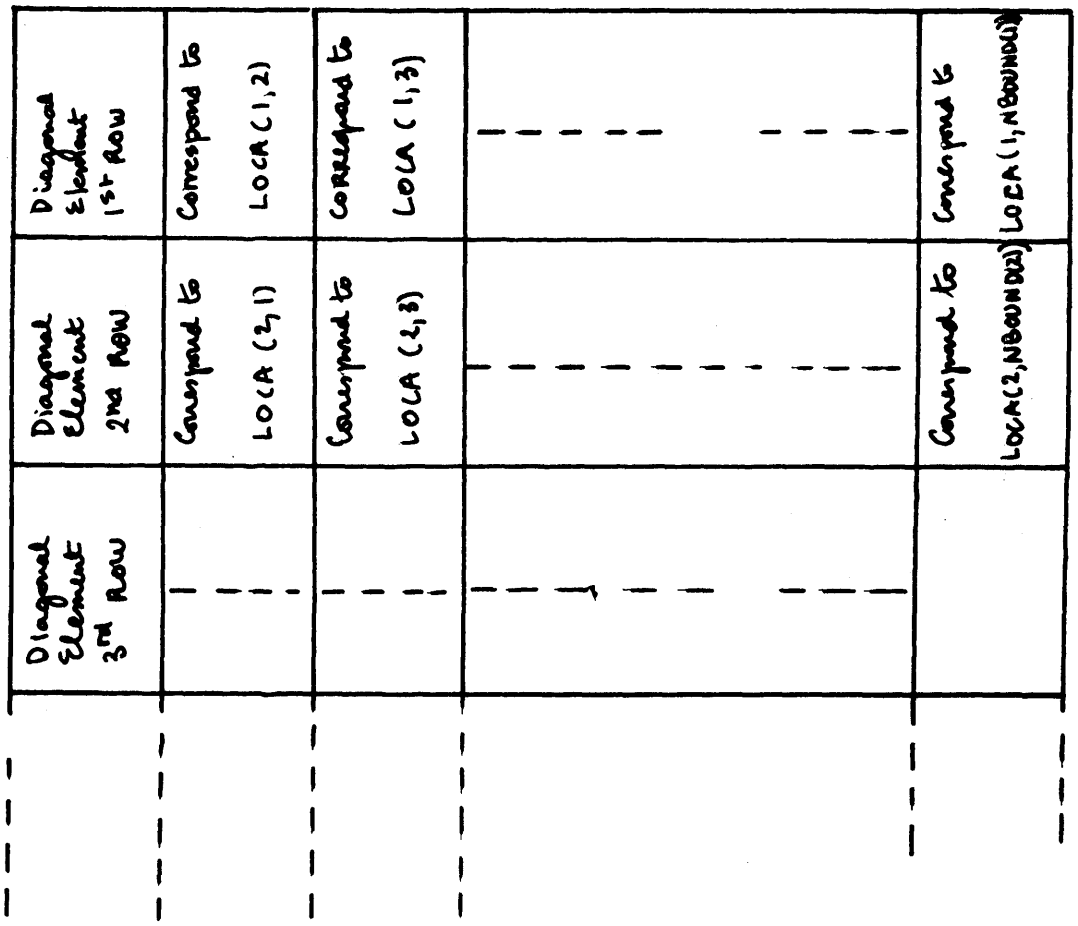


TABLE B.2: Print-Out from DIVERT of the Array A

:EW COMPACT AAA:

1.3234E+01	6.4054E+00	-6.4059E+00
1.3235E+01	-6.4060E+00	6.4059E+00
1.9636E+01	-6.4059E+00	6.4059E+00
1.3233E+01	6.4052E+00	-6.4042E+00
1.3235E+01	-6.4060E+00	6.4059E+00
1.9642E+01	-6.4059E+00	6.4061E+00
1.3234E+01	-6.4042E+00	6.4045E+00
1.9643E+01	-6.4063E+00	-6.4064E+00
1.3229E+01	-6.4042E+00	6.4042E+00
1.9642E+01	-6.4063E+00	-6.4064E+00
1.9638E+01	-6.4019E+00	6.4019E+00
1.9632E+01	-6.4019E+00	6.4017E+00

-6.4059E+00	0.0	0.0	0.0
-6.4061E+00	-6.4059E+00	0.0	0.0
0.0	0.0	0.0	0.0
-6.4045E+00	0.0	0.0	0.0
6.4064E+00	-6.4062E+00	-6.4063E+00	0.0
1.2813E+01	0.0	0.0	0.0
6.4063E+00	-6.4062E+00	-6.4063E+00	0.0
6.4063E+00	1.2814E+01	0.0	0.0
-6.4017E+00	-6.4019E+00	0.0	0.0
6.4062E+00	1.2813E+01	0.0	0.0
1.2812E+01	0.0	0.0	0.0
0.0	0.0	0.0	0.0

This is more than useful for the programming of the iterative procedure, since the element of AAA and those of each cross-flow vector are related through LOCA for their value, position and sign.

TABLE B. 3: Print-Out from ITER of the Array A

EW COMPACT AAA:

1.3234E+01	6.4054E+00	-6.4059E+00	
1.3235E+01	-6.4060E+00	6.4059E+00	
1.9636E+01	-6.4059E+00	6.4059E+00	
1.3233E+01	6.4052E+00	-6.4042E+00	
1.3235E+01	-6.4060E+00	6.4059E+00	
1.9642E+01	-6.4059E+00	6.4061E+00	
1.3234E+01	-6.4042E+00	6.4045E+00	
1.9643E+01	-6.4063E+00	-6.4064E+00	
1.3229E+01	-6.4042E+00	6.4042E+00	
1.9642E+01	-6.4063E+00	-6.4064E+00	
1.9638E+01	-6.4019E+00	6.4019E+00	
1.9632E+01	-6.4019E+00	6.4017E+00	

-6.4059E+00	0.0	0.0	0.0
-6.4061E+00	-6.4059E+00	0.0	0.0
0.0	0.0	0.0	0.0
-6.4045E+00	0.0	0.0	0.0
6.4064E+00	-6.4062E+00	-6.4063E+00	0.0
1.2813E+01	0.0	0.0	0.0
6.4063E+00	-6.4062E+00	-6.4063E+00	0.0
6.4063E+00	1.2814E+01	0.0	0.0
-6.4017E+00	-6.4019E+00	0.0	0.0
6.4062E+00	1.2813E+01	0.0	0.0
1.2812E+01	0.0	0.0	0.0

TABLE B.4: Typical Input for a 10 Subchannel Case

```
                                                                                                                    C4
   10    15     5    10    21
 2000
1    1     0          PWR    NC=10     STRIPE BOUNDARY NUMBERING
     1    -1
     1    1000.0    2600.0      30
     2     1     0     0     0
0.184 -0.2
     3    21
0.0  0.144 0.050.309 0.100.783 0.151.050 0.201.280 0.251.45
 0.301.54  0.351.59  0.401.60  0.451.56  0.501.50  0.551.40
 0.601.28  0.651.14  0.700.95  0.750.775 0.800.588 0.850.450
 0.900.312 0.950.200 1.000.094
     4
    2  10    7    3    4    1    0    3
1    1        1.0     37.43     348.0     310.2     0.122       0.0     0.374     264.0
4.0110.9780.565
1    7        0.5     37.43     348.0     310.2     0.122       0.0     0.374     264.0
4.0110.9780.565
1.1281.0901.1521.0031.1651.1521.0631.1641.1821.068
0.05      1 .16      2 .32      3 .49      3 .66      3 .87      2 .99      1
```

```
    7    8    9   10   -1
    0    1    3    6   10
    0    2    5    9
    0    4    8
    0    7
    0

  2.0   .08 640..3225 8.80 .076405.0.02251000.
     9     1     1     0
    .5    0. 144.    0.    10     1    1.    20
    10     1     0     0     1
 0.02
    11     1     1     2     2     2     2
      2250.       557.3       2.66        0.189
  0.0   1.0    1.1.056
  0.0   1.0    1.1.15
   0.1.000    1. 0.77
   0.1.000    1.1.000
   12     3     2     2     3
 110
 110
 1 4 5
```

B. 3 ITERATIVE METHOD FOR A MODIFIED COBRA -III-C

The programming of the SOR method is straightforward once the new set-up of the array AAA is established. It has seemed useful however, to present the Flow Chart, Fig.B. 1 .

Note first that the part of ITER dealing with the calculation of the cross-flow is not a subroutine - as DECOMP and SOLVE were for DIVERT. The basic idea is to have a very flexible programming in which any combination of constants, useful for the calculation, or even the convergence criteria can be varied without difficulty. The programming can be divided into several steps.

1 - Set-up of the constants,

2 - Initialisation of cross-flow,

3 - Iterative cross-flow calculation,

4 - Test with convergency criteria, and

5 - Print-out of the results.

1. Set-up of Constants

```
EPS = 0.01 x FERROR
OMEGA = 4./3.
```

Note that one can change, according to the desired accuracy of the results, the value of EPS - as mentioned before.

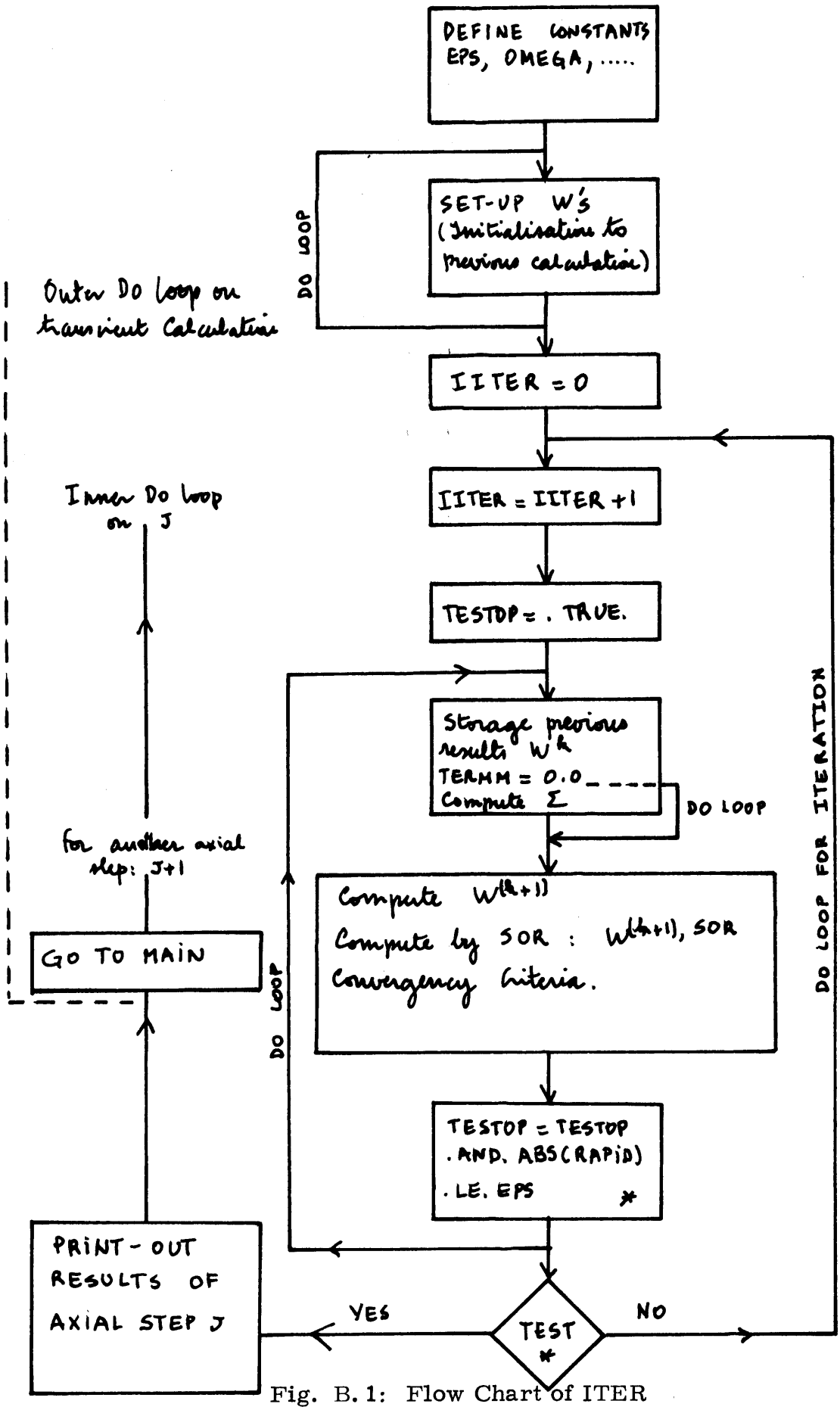


Fig. B. 1: Flow Chart of ITER

2. Initialization of Cross-Flows

```
(1)     DO 1986 KL = 1, NK
(2)     DATA ($ANSWE + KL) = DATA($W + KL + MG*(J-1))
(3) 1986 CONTINUE
```

At J=1, the cross-flow value is initialized for the steady state case to zero.

At J = j, the cross-flow value is initialized, before iteration, to the results of the previous axial step J=j-1. This initialization is intended to hasten the convergence process.

Note that for each axial step J, only NK cross-flow have to be calculated. The iteration procedure must be understood as being carried out until the convergency criterion is satisfied. The value of J - the axial plan on which the calculation is done - being determined in MAIN.

For clearer understanding, it has been decided to call by vector cross-flow, $\underline{W}_J$, the NK set of cross-flows in a same axial step: therefore, each cross-flow can be considered as a component of $\underline{W}_J$. The calculation procedure starts with the value $\underline{W}_{J-1}$ as an initial value, and ends up, when the test is satisfied, with $\underline{W}_J$.

Note that for the steady state case every cross-flow vector in each axial step is set to 0. At the end of the steady calculation one has $\underline{W}_1^{SOR}$ $\underline{W}_2^{SOR}$, For the transient state case the initial values of the components of the cross-flow vector are: W, W_2

At the end of the first calculation - for the transient - one ends up with: $\underline{W}_{1,T1}^{SOR}$, $W_{2,T1}^{SOR}$ -----
the subscript being for the first run of the transient calculation which requires seven successive runs.

Generally one gets: initial value for 1th step of transient calculation: $\underline{W}^{SOR}_{1,T_{\ell-1}}$, $\underline{W}^{SOR}_{2,T_{\ell-1}}$, ...

and ends up with: $\underline{W}^{SOR}_{1,T_{\ell}}$, $\underline{W}^{SOR}_{2,T_{\ell}}$, ...

3. Iterative Cross-Flows Calculation Test with Convergency Criteria

```
      IITER = 0                                                              (4)
2008  IITER = IITER + 1
      TESTOP = .TRUE.
      DO 2001 MMK = 1,NK
      RESERV = DATA ( $ANSWE + MMK)
      TERMM = 0.0
      DO 2002 NTR = 2, 7
      INDI = IDAT ( $LOCA + MMK + MG * (NTR-1))
      INDI = IABS (INDI)
      TERMM = TERMM + DATA ($AAA + MMK + NK * (NTR-1)) * DATA ($ANSWE+INDI)
2002  CONTINUE
      DATA ($ANSWE + MMK) =(DATA ($B + MMK) - TERMM) / (DATA ( $AAA + MMK))
      DATA ($ANSWE+MMK) = (1.0 - OMEGA) * RESERV + OMEGA * DATA($ANSWE + MMK)  (15)
      RAPID = (RESERV - DATA ($ANSWE +MMK)) / ( RESERV
      TESTOP= TESTOP . AND. (ABS (RAPID). LE. EPS)
2001  CONTINUE
      IF ((.NOT. TESTOP). AND. (IITER. LE. 200)) GO TO 2008
                                                                             (9)
```

IITER, after being initialized to zero, for each axial step, is set up at 2000 to IITER = IITER + 1:

Note that when the cross-flow vector $\underline{W}^{SOR}_{\ell,J}$ is calculated - i.e. when each component $1 \leqslant i \leqslant NK$ satisfies the convergence criteria - then IITER indicates the number of iterations which has been required for the vector calculation and note for each component only.

If this number is larger than 200 - i.e. the iteration procedure is by far too long - the calculation is stopped. This is intended to protect the calculation time.

TESTOP is initialized to . TRUE. and the print-out of the results will be done if and only if . TESTOP. is set up to . TRUE. after the iterations. If it is set up on FALSE then the calculation is carried out again starting at 20008.

The DO Loop on MMK with (1 MMK NK) is intended to store the previous computed value of the cross-flow components.

TERMM is, of course, set to 0 for each component before calculation. Then, DO Loop on NTR (2, 7) is the sum of the products of the cross-flow coefficients with the corresponding cross-flow component. Note that this important expression is rather straightforward with a system of indices - INDI - directly related to LOCA.

DATA (AAA + MMK + NK * (NRT - 1)) is the cross-flow coefficient value associated to the cross-flow component value DATA (AAA + INDI).

It has seemed useful to detail a feature of the coding dealing with the dynamic storage of the results. Instead of using two NK storages - one for the components at kth iteration and one for the components at (k + 1) th iteration - only one NK storage is used. It consists of storing successively an NK storage containing the components of then the k th iteration with components of the (k + 1) th iteration. Fig. 4 summarizes this procedure.

Once the SOR procedure is applied, note that the test is carried out. If the test is satisfied for the i th component, the calculation continues in the DO LOOP (2001) with the (i + 1) th component. (1 i NK)

The test value - TESTOP - must be set up to TRUE after the calculation and test of <u>all</u> components in order to print out the results. If the value of the test is set up to - FALSE - i. e. if one of the components does not satisfy the convergency criteria - the whole process - i. e. iteration of the component - is carried out again until the test value <u>KEEPS FOR ALL COMPONENTS</u>, the value . TRUE.

```
      WRITE (6,2090) IITER, J
2090  FORMAT (/,7H IITER=,I3, 5X, 12H AXIAL STEP=,I3)
      DO 150 K=1,NKK
 150  DATA ($W+K+MG* (J-1)) = DATA($ANSWE+K)
      RETURN
      END
```

Once the convergence criterion is satisfied, the results can be printed out. Note that the results are printed out at each axial step J.

Fig B2: Programming Details of ITER.

INITIALIZATION BY (1), (2), (3)

DATA ($ANSWE + KL) = DATA ($W + KL + MG * (J-1))

$1 \leq KL \leq NK$

STORAGE OF PREVIOUS CALCULATED COMPONENT AT STEP (J-1)

* NOTE: FOR J = 1, all W's are zero.

Computation of Cross-Flow Component - for each non-flow vector.

Assume k^{th} iteration

STEP 1 : Compute W_1^k

$W_1^{(k-1),SOR}, W_2^{(k-1),SOR}, \ldots\ldots, W_{NK}^{(k-1),SOR}$ *

STORAGE OF THE ARRAY BEFORE CALCULATION, AT ITERATION K

Formula is :

$$W_1^{(k)} = \frac{1}{a_{11}} \left(b_1 - \sum_{\ell=2}^{NK} a_{1\ell} W_\ell^{(k-1)} \right)$$

DONE BY

MMK = 1

RESERV = DATA ($ANSWE + MMK)

ie : $W_1^{(k-1)}$

TERMM = TERMM + DATA ($AAA + MMK + NK * (NTR-1)) * DATA ($ANSWE + INDI)

DATA ($ANSWE + MMK) = DATA ($B + MMK) - TERMM / (DATA ($AAA + MMK))

+ FORMULA FOR S.O.R.

STORAGE BECOMES

IF CRITERIA SATISFIED THEN

$W_1^{(k),SOR}, W_2^{(k-1),SOR}, \ldots\ldots, W_{NK}^{(k-1),SOR}$

GENERALLY FOR COMPUTING

JUST BEFORE COMPUTATION OF ℓ^{th} COMPONENT STORAGE WILL BE

$W_1^{(k),SOR}, W_2^{(k),SOR}, \ldots, W_{\ell-1}^{(k),SOR}, \ldots W_{NK}^{(k-1),SOR}$

This is still : $W_\ell^{(k-1),SOR}$

AFTER ITERATION K :

$W_1^{(k),SOR}, \ldots\ldots, W_\ell^{(k),SOR}, \ldots, W_{NK}^{(k),SOR}$

- ie if

$|W_\ell^{(k),SOR} - W_\ell^{(k-1),SOR}| \leq \varepsilon$

for all $1 \leq \ell \leq NK$

$W_\ell^{(k),SOR}$ for all $1 \leq \ell < NK$

WITH THIS DYNAMIC STORAGE, (15) IS COMPUTED WITHOUT USING 2 SYSTEMS OF INDICES $\Rightarrow$ 2 ARRAYS.

$$W_\ell^{(k)} = \frac{1}{a_{\ell\ell}} \left(b_\ell - \sum_{i=1}^{\ell-1} a_{i\ell} W_i^{(k)} - \sum_{i=\ell+1}^{NK} a_{i\ell} W_i^{(k+1)} \right)$$

TABLE B. 4: Typical AAA for a 10 Subchannel Case

```
:EW COMPACT AAA:
    1.3234E+01     6.4054E+00    -6.4059E+00
    1.3235E+01    -6.4060E+00     6.4059E+00
    1.9636E+01    -6.4059E+00     6.4059E+00
    1.3233E+01     6.4052E+00    -6.4042E+00
    1.3235E+01    -6.4060E+00     6.4059E+00
    1.9642E+01    -6.4059E+00     6.4061E+00
    1.3234E+01    -6.4042E+00     6.4045E+00
    1.9643E+01    -6.4063E+00    -6.4064E+00
    1.3229E+01    -6.4042E+00     6.4042E+00
    1.9642E+01    -6.4063E+00    -6.4064E+00
    1.9638E+01    -6.4019E+00     6.4019E+00
    1.9632E+01    -6.4019E+00     6.4017E+00

-6.4059E+00     0.0             0.0             0.
-6.4061E+00    -6.4059E+00      0.0             0.
 0.0            0.0             0.0             0.
-6.4045E+00     0.0             0.0             0.
 6.4064E+00    -6.4062E+00     -6.4063E+00      0.
 1.2813E+01     0.0             0.0             0.
 6.4063E+00    -6.4062E+00     -6.4063E+00      0.
 6.4063E+00     1.2814E+01      0.0             0.
-6.4017E+00    -6.4019E+00      0.0             0.
 6.4062E+00     1.2813E+01      0.0             0.
 1.2812E+01     0.0             0.0             0.
```

B.4 CONCLUSIONS AND RECOMMENDATIONS

An overall comparison between the calculation time for the two methods - Gauss Elimination Method and Successive over Relaxation - can now be envisionned.

For small order matrices, the iterative solution computational time is larger by a factor of two than the Gauss Elimination method. When larger cases are considered the calculation time reduces significantly from 35% for the 128 case to an estimated 80% for the 356 case.

Note that this result is stated conservatively. Favorable matrix properties can improve the relative advantages of the Gauss-Siedel method. If the matrice happens to be diagonally dominant, the calculation time will be reduced for any order by a factor of four.

If a theoretical estimate of the calculation time should be given, the formula proposed by Bowring[3] could be used:

$$t = 0.00282NK + 0.000000837 \times NK \times MS^2 + 0.0000127\ NK^2 \qquad (8)$$

The first term accounts for the set-up of equations. With each solution method it is identical: 0.00282 NK.

The second term represents the calculation time for the storage of the matrix. For iterative method MS is fixed to be 7; therefore this term amounts to 0.000 0005859 NK, and the last term is estimated to be proportional to NK^2 - for the Gaussian Elimination, when the theoretical estimate gives a calculation time proportional to NK^3. Therefore, this estimate is believed to be too optimistic for the direct method. The calculation time for the Successive over Relaxation method cannot be smaller than proportional to NK^2 for each

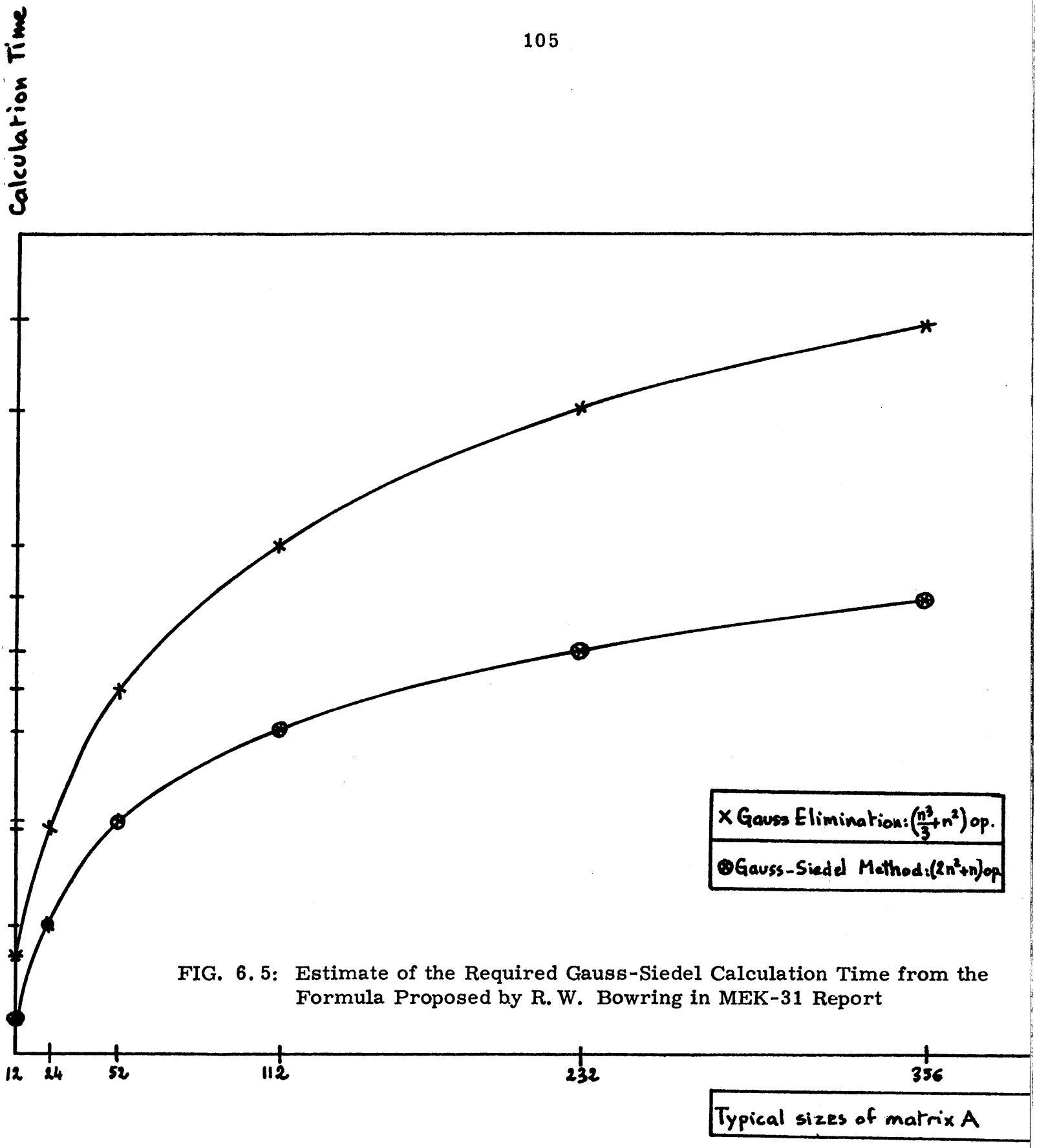


FIG. 6.5: Estimate of the Required Gauss-Siedel Calculation Time from the Formula Proposed by R. W. Bowring in MEK-31 Report

Fig. B.3: Calculation Time Estimate

iteration. An average of 14 iterations is required for acceptably accurate results. Therefore, the last term of Eq. can be written: $0.0000\ 127 \times 14 \times NK^2 = 0.0001778\ NK^2$.

(9)

The formula becomes:

$$0.00282\ NK + 0.000\ 005859\ NK + 0.000\ 1778\ NK^2 \qquad (10)$$

This theoretical estimate should give a rather rough idea for computing larger matrices.

REFERENCES

1. R. W. Bowring, "An Introduction to COBRA-III-C Coding", MEK report series, 20, (1974).

2. R. W. Bowring, "The Coding for a Faster COBRA-III-C Cross-Flow Solution", MEK report series, 28, (1975).

3. R. W. Bowring, "Timing Runs for the Original and Modified COBRA-III-C", MEK report series, 31, (1975).

4. R. W. Bowring, "MEKIN - Type Input Data Presentation in COBRA-III-C, MEK report series, 52, (1975).

5. V. N. Faddeeva, "Computational Method of Linear Algebra", Dover Publications Inc. (1959).

6. J. M. Ortega, "Numerical Analysis - a 2nd Course", Academic Press (1972).

7. J. Wilkinson, "The Algebraic Eigenvalue Problem", Clarendon University Press, (1965).

8. C. Moler and G. Forsythe, "Computer Solution of Linear Algebraic System", Prentice-Hall, (1967).

9. M. Clark Jr. and K. F. Hansen, "Numerical Methods of Reactor Analysis", Academic Press, (1964).

10. K. R. James and W. Riha, "Convergence Criteria for Successive over Relaxation", SIAM Journal of Numerical Analysis, 12, 2, (1975).

11. C. F. Van Loan, "A General Matrix Eigenvalue Algorithm", SIAM Journal of Numerical Analysis, 12, 6, (1975).

12. M. M. Blevins and G. W. Stewart, "Calculating Eigenvectors of Diagonally Dominant Matrices", Journal of the Association for Computer Machinery, 21, 2, (1974).

13. A. M. Ostrowski, "Solution of Equations and Systems of Equations", Academic Press, (1960).

14. B. A. Chartres and J. Geulder, "Computable Error Bounds for Direct Solution of Linear Equations", Journal of the Association for Computer Machinery, 14, 1, (1967).

Other Useful References

M. C. Pease, "Matrix Inversion Using Parallel Processing", Journal of the Association for Computer Machinery, 14, 4, (1967).

H. J. Bowdler, R. S. Martin, G. Peters and J. H. Wilkinson, "The Solution of Real and Complex System of Linear Equations", Numerichke Mathematik, 8, (1966).

F. G. Gustavson, W. Liniger and R. Willoughby, "Symbolic Generation of an Optimal Crout Algorithm for Sparse System of Linear Equations", Journal of the Association for Computer Machinery, 17, 1, (1970).

G. Zielke, "Inversion of Modified Symmetric Matrices", Journal of the Association for Computer Machinery, 15, 3, (1968).