THE MULTIDIMENSIONAL WEGSTEIN METHOD AND THE COMPARISON OF CONVERGENCE PROCEDURES

by

HARRY CHARLES FRANK

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF BACHELOR OF
SCIENCE

at the
Massachusetts Institute of
Technology

June, 1976

Signature of Author Signature redacted ....
Department of Chemical Engineering
1976

Certified by Signature redacted ....
Thesis Supervisor

Accepted by Signature redacted ....
Chairman, Department

ABSTRACT

This paper reports the results of ny investigations as to the advantages of the MultidimensionslWegstein Method over the Classical Method of Successive Substitution, the Bounded Wegstein Method, and the Newton-Raphson Method. I demonstrated its superiority by applying them to several engineering example problems, and to Mathematical model problems.

The method solved all the engineering problems better than any of the other methods. This method does require more time for each iteration, so its speed must be balanced against this difference.

The Mathematical problems were chosen such that the procedures performance for various values of the parameters would relate information as to the classes of problems the various iterative procedures are likely to solve well. These results suggest that the Multidimensional Wegstein method is likely to solve a very wide class of functions, and to do so in less function evaluations than the other methods.

OUTLINE

I. Introduction p. 6

II. Problem Formulation p. 9

(A) General Definition
(B) Method Descriptions

III. Engineering Examples p. 21

(A) Aluminum Purification Problem
(B) Combustion Example
(C) Ethylene Dichloride Reaction
(D) Photochemical Reaction Problem
(E) Nagiev Oil Separation Problem

IV. Mathematical Examples p. 48

(A) Linear Variables, Linear Interaction
(B) Linear Variables, Non-Linear Interaction
(C) Non-Linear Variables, Linear Interaction

V. Conclusions p. 88

VI. Suggestions for further study p. 93

VII. Footnotes p. 94

VIII. Appendices p. 95

LIST OF FIGURES

Fig. 1	Flow Sheet, Successive Subst.	p. 12
Fig. 2	Flow Sheet, Newton-Raph. Meth.	p. 14
Fig. 3	Flow Sheet, Bounded Weg. Meth.	p. 17
Fig. 4	Flow Sheet, Multi. Weg. Meth.	p. 20
Fig. 5	Aluminum Purification Set-up	p. 22
Fig. 6	Alumin. Purif. Prog. Listing	p. 25
Fig. 7	Alum. Purif. Prob., Press=1	p. 27
Fig. 8	Alumin. Pur. Prob., Pres= 3.0	p. 28
Fig. 9	Alum. Purif. Prob., Press= 5.0	p. 29
Fig. 10	Alum. Purif. Prob., Press=38.0	p. 30
Fig. 11	Combustion Prob. Listing	p. 33
Fig. 12	Combustion Prob., Press=1.0	p. 35
Fig. 13	Combustion Prob., Press=32.0	p. 36
Fig. 14	Ethylene Dichloride System	p. 38
Fig. 15	Ethylene Dichloride Listing	p. 39
Fig. 16	Eth. Dichloride Prob.,A=B=45	p. 40
Fig. 17	Eth. Dichloride Prob.,A=B=50	p. 41
Fig. 18	Photochemical Prob. Listing	p. 42
Fig. 19	Photochemical Prob. Sensitiv.	p. 44
Fig. 20	Oil Separation Prob. Listing	p. 46
Fig. 21	Oil Separation Prob. Sens.	p. 47

LIST OF FIGURES(CONT'D)

Fig. 22	Lin.Var./Lin Interact. "spot"check	p. 50
Fig. 23	Lin.Var./Lin Interact. "spot"check	p. 50
Fig. 24	Lin.Var./Lin Interact. "spot"check	p. 50
Fig. 25	Lin.Var./Lin Interact. B= 0.0	p. 51
Fig. 26	Lin Var./Lin Interact. B=0.01	p. 52
Fig. 27	Lin.Var./Lin Interact. B=0.1	p. 53
Fig. 28	Lin. Var./Lin Interact. B=0.2	p. 54
Fig. 29	Lin.Var./Lin Interact. B= 0.4	p. 55
Fig. 30	Lin.Var./Lin Interact. B= 1.0	p. 56
Fig. 31	Lin.Var./Lin Inter. B= 200,800	p. 57
Fig. 32	Lin.Var./Lin Int "spot" check, A	p. 61
Fig. 33	Lin.Var./Lin Int Prob., A=1.25$^{10}_{10}$-6	p. 62
Fig. 34	Lin.Var./Lin Int Prob., A=0.5	p. 63
Fig. 35	Lin.Var./Lin Int. Prob., A= 0.7	p. 64
Fig. 36	Lin.Var./Lin Int. Prob., A=0.99	p. 65
Fig. 37	Lin.Var./Lin Int., A= 5.0, 100	p. 66
Fig. 38	Lin.Var./Non-lin. Int. "spot"check	p. 68
Fig. 39	Lin.Var./Non-lin. Int. "spot"check	p. 68
Fig. 40	Lin.Var./Non-lin. Int."spot"check	p. 68
Fig. 41	Lin.Var./Non-lin Int., B=0.0	p. 70
Fig. 42	Lin.Var./Non-lin Int., B=0.01	p. 71

LIST OF FIGURES(CONT'D)

Fig. 43	Lin.Var./Non-lin. Int., B=.011	p. 72
Fig. 44	Lin.Var./Non-lin. Int., B=.0125	p. 73
Fig. 45	Lin.Var./Non-lin. Int., B=-.01	p. 74
Fig. 46	Lin Var./Non-lin Int.,B=-1,-50	p. 75
Fig. 47	Lin.Var./Non-lin Int., "spot"check	p. 77
Fig. 48	Lin. Var./Non-lin Int., A=-0.5	p. 78
Fig. 49	Lin. Var./Non-lin Int.,A=5,100	p. 79
Fig. 50	Non-lin.Var./Lin Int. "spot" check	p. 81
Fig. 51	Non-lin.Var./Lin Int. "spot"check	p. 81
Fig. 52	Non-lin.Var./Lin Int. Sens. B	p. 82
Fig. 53	Non-lin.Var./Lin Int., A=0.0	p. 85
Fig. 54	Non-lin. Var./Lin Int., A=0.02	p. 86
Fig. 55	Non-lin.Var./Lin Int., A=0.04	p. 87

I. INTRODUCTION

The development of new processes, the evaluation of alternative plant designs and the improvement of existing processes rely on the mathematical modeling of the process itself, and usually require the computer aided simulation of the process. The most widely accepted approach to this problem has been the "modular" formulation-- a package of "unit computational subroutines" simulate each particular piece of equipment in a plant design. These units are connected to each other by an executive program, which calls each computational routine in sequence, using the product stream from one unit as the feed stream to the next unit in the plant model.

When no recycle streams are present, and the feed condition to the first unit are all known, the steady-state process simulation can be effected directly by one sequential pass through each of the units. When a recycle stream is present, however, one must guess its stream values and employ some sort of iterative procedure. The executive Routine, therefore, must handle the processing of all input and output information, the flow of information between units and the convergence of all recycle streams in the process formulation. The "unit computational routines", on the other hand, are models of processing equipment, which can be used for a variety of process simulations (any simulation requiring that particular piece of equipment).

The formulation of executive subprograms and unit computational routines has been extensively researched[1] and currently are in general use. The choice of an iterative convergence procedure to use, however, is still a topic of research. Many procedures have been developed in recent years, that appear to be better than the classical method of Successive Substitution. One of the purposes of this work is to present one more procedure (the Multidimensional Wegstein Method), that appears to solve a wider variety of recycle problems than the other methods, and does so in fewer function evaluations. The method would be much more efficient for problems which require a lot of time to evaluate the functions, like plant simulations.

Previously, convergence procedures were evaluated by applying them to several engineering example problems, and by applying a few other methods to the same problem-- they are compared as to the rate at which they converge to the solutions (data is recorded in error versus iterations plots). I have done this with the Multidimensional Wegstein Method: comparing it to the Newton-Raphson Technique, the Method of Successive Substitution, and the Bounded Wegstein Method for a variety of engineering examples.

The other purpose of this paper is to suggest a more general method for evaluating the many methods of converging recycle streams in order to develop an understanding of when the method on hand is most likely to be the quickest method for solving any particular type of problem. Hopefully, sweeping generalizations will be easily formulatable(e.g. Highly interacting problems are solved much more quickly by the Multidimensional Wegstein Method).

II. PROBLEM FORMULATION

A: General Definition:

In order to compare the efficiencies of the four methods, I programmed subroutines which would predict the values of the variables based upon each of the methods, Successive Substitution, the Newton-Raphson Method, the Bounded Wegstein Method, and the Multidimensional Wegstein Method. A list of the auxillary subroutines used to effect this goal can be found in Appendix A, while the listings for the methods themselves, along with a sample solution can be found in appendix B. Subroutine XQT9 initializes the starting guesses, then calls each of the methods to the problem at hand. A different Subroutine FUNV9 simulates each of these problems. (see Section III, Engineering Examples).

I programmed five different Chemical Engineering example problems, and obtained solutions (plots of error vs. function evaluations) by each of the four methods. In order to get a feel for how varying degrees of interaction affect each method, I investigated several actual engineering example problems.

Of the many possible ways to define the error of the system, I chose to consider each variable separately and consider the error of the system to be the maximum error amongst all the variables. The individual variable's error was taken to be the minimum of the absolute difference between the variable and its function value and the relative error, the absolute error divided by the variable's value.

In order to obtain a deeper understanding of the variables that affect the rate of obtaining solutions, I programmed several simple Mathematical cases. One of the cases I investigated was that of linear variables with linear interaction. The particular set of equations employed was:

$$x_1 = Ax_1 + Bx_2 + C$$
$$x_2 = Ax_2 + Bx_1 + C$$

By varying the value of B, one alters the amount of interaction amongst the two variables. For example, if B=0, there would be no interaction present, but for any finite value, there is a finite amount of interaction. This problem is particularly important for, near the solution, all problems appear linear. Varying the value of A, affects the degree to which the value of x depends upon its own value, intra-variable interactions.

Many values for the variables of this example were solved in order to obtain as clear an idea of the affects of these values as possible.

A few runs were also attempted, varying the exponents on the unknowns in order to see the affects of these changes. For the results of these runs, see Section IV, Mathematical Examples.

B: Method Descriptions:

(i) The Classical Method of Successive Substitution is one of the simplest methods to understand, and to implement. Each equation must be solved for a single output variable to obtain the form:

$$\underline{x} = F(\underline{x}) \qquad (1)$$

An initial assumption, $\underline{x}^{(0)}$, is made and then successive improvements are attempted by applying the algorithm:

$$\underline{x}^{(k+1)} = F(\underline{x}^{(k)}) \qquad (2)$$

The flow sheet for this method is illustrated in Fig. 1, while the subroutine listing and example problem are listed in Appendix B.

(ii) The Newton-Raphson Method proceeds by expanding the set of equations $F(\underline{x})=0$ in a Taylor Series about the present guess, $\underline{x}=\underline{x}^{(k)}$, and retaining only the linear terms:

$$\begin{aligned} f_1 &= f_1\big|_k + \frac{df_1}{dx_1}\bigg|_k dx_1 + \dots \frac{df_n}{dx_n}\bigg|_k dx_n \\ &\vdots \\ f_n &= f_n\big|_k + \frac{df_n}{dx_1}\bigg|_k dx_1 + \dots \frac{df_n}{dx_n}\bigg|_k dx_n \end{aligned} \qquad (3)$$

where $dx_i = x_i - x_i^k$, and $|$ means evaluated at $\underline{x} = \underline{x}^{(k)}$.

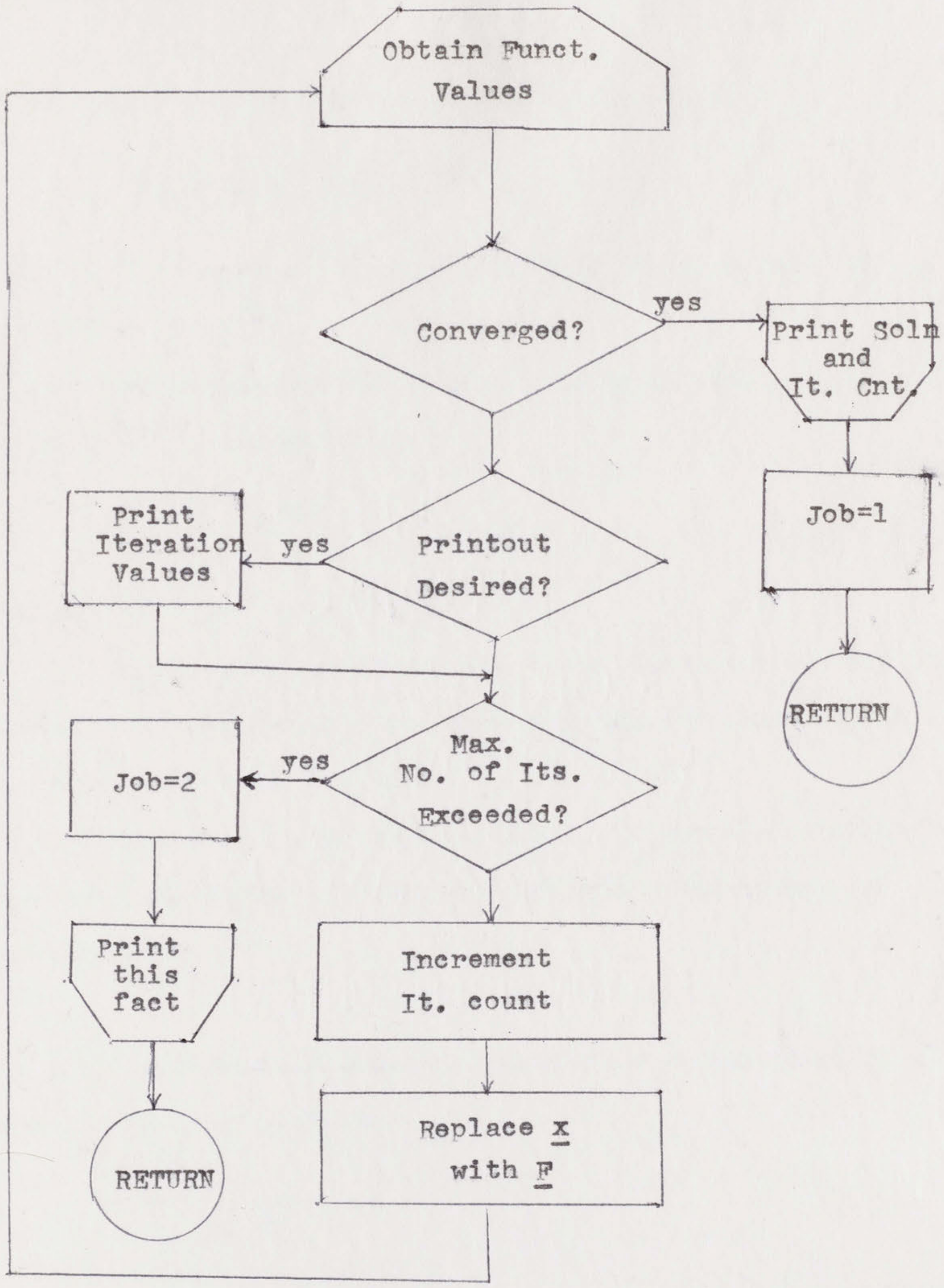


FIGURE 1 SUCCESSIVE SUBSTITUTION

This system of equations can be represented as:

$$\underline{f}(\underline{x}) = \underline{f}(\underline{x}^{(k)}) = J(\underline{x}^{(k)})d\underline{x} = 0 \qquad (4)$$

where $J(\underline{x}^{(k)})$ denotes the Jacobian of $\underline{f}$ with respect to $\underline{x}$, evaluated at $\underline{x} = \underline{x}^{(k)}$.

The Newton-Raphson Technique generates the new approximation $\underline{x}^{(k+1)}$, according to:

$$\underline{x}^{(k+1)} = x^{(k)} + d\underline{x} \qquad (5)$$

where $d\underline{x} = -J^{-1}(\underline{x}^{(k)})f(\underline{x}^{(k)})$

The Jacobian is acquired for each iteration by perturbing each variable separately, and calculating the derivatives with respect to each variable.

The flow sheet for this method is illustrated in Fig. 2, while the Subroutine listing and example problem are in Appendix B.

(iii) The Bounded Wegstein Method is again formulated to solve the recycle problem:

$$\underline{x} = F(\underline{x}) \qquad (1)$$

where $F(\underline{x})$ is the recycle solution for the variables $\underline{x}$ in the simulation problem.

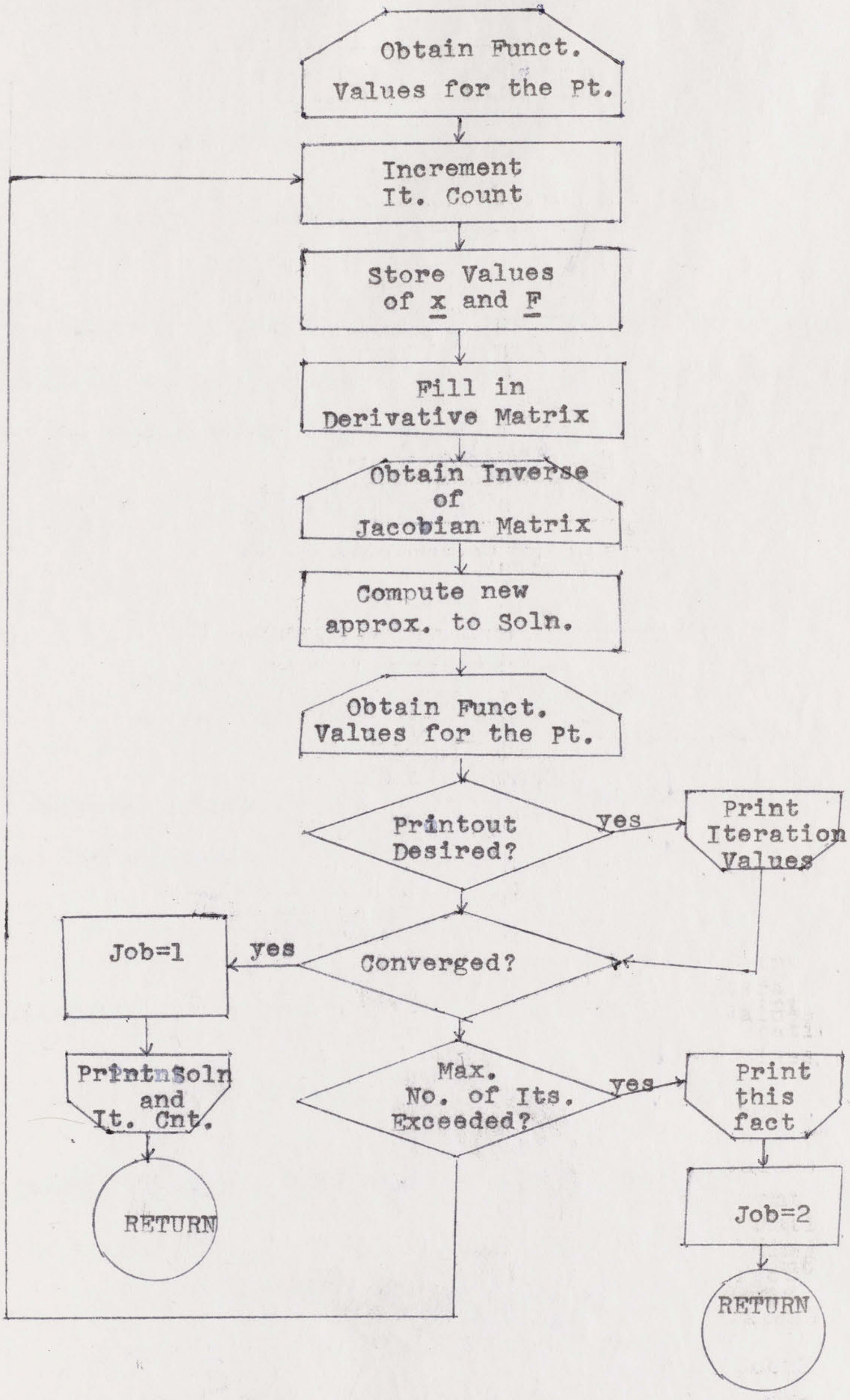

Obtain Funct. Values for the Pt.
Increment It. Count
Store Values of x and F
Fill in Derivative Matrix
Obtain Inverse of Jacobian Matrix
Compute new approx. to Soln.
Obtain Funct. Values for the Pt.
Printout Desired?
yes
Print Iteration Values
Converged?
yes
Job=1
Print Soln and It. Cnt.
RETURN
Max. No. of Its. Exceeded?
yes
Print this fact
Job=2
RETURN

FIGURE 2 SUBROUTINE NEWR9

The Wegstein Method assumes a linear form for $F(\underline{x})$:

$$F(\underline{x}) = a\underline{x} + b \tag{6}$$

The parameters, a and b, are obtained from two sets of $(\underline{x}, F(\underline{x}))$, obtained by successive substitution. Using these two $(\underline{x}, F(\underline{x}))$ pairs the equation above is solved for a, and b.

$$F^{(0)} = ax^{(0)} + b$$
$$F^{(1)} = ax^{(1)} + b$$

$$\tag{7}$$

or:

$$a = (F^{(1)} - F^{(0)})/(x^{(1)} - x^{(0)})$$
$$b = (F^{(0)}x^{(1)} - F^{(1)}x^{(0)})/(x^{(1)} - x^{(0)})$$

The next guess is the value of $\underline{x}$, for which the assumed form of F(x) equals x, or:

$$x^{(2)} = ax^{(2)} + b \tag{8}$$

therefore,

$$x^{(2)} = b/(1-a) \tag{9}$$

by substituting the solutions for a and b we obtain:

$$x^{(2)} = \frac{F^{(0)}x^{(1)} - F^{(1)}x^{(0)}}{(x^{(1)} - x^{(0)}) - (F^{(1)} - F^{(0)})} \tag{10}$$

By defining a convergence parameter q, such that:

$$x^{(2)} = qx^{(1)} + (1-q)F^{(1)} \tag{11}$$

solving these equations for q yields:

$$q = \frac{F^{(1)} - F^{(0)}}{(F^{(1)} - F^{(0)}) - (x^{(1)} - x^{(0)})} \tag{12}$$

or in terms of a:

$$q = a/(a-1) \tag{13}$$

generalizing these results for the uni-variable case:

$$\begin{gathered} x^{(i+1)} = qx^{(i)} + (1-q)F^{(i)} \\ q = a/(a-1) \\ a = df/dx \end{gathered} \tag{14}$$

where df, and dx are defined as:

$$\begin{aligned} df &= F^{(i)} - F^{(i-1)} \\ dx &= x^{(i)} - x^{(i-1)} \end{aligned} \tag{15}$$

Subroutine Bweg9 was formulated to apply this procedure independently to each variable of multi-variable problems. The parameter q was bounded in the range $-20 \leq q \leq 0$. The flow sheet for this method may be found in Fig. 3, while the subroutine listing is in Appendix B.

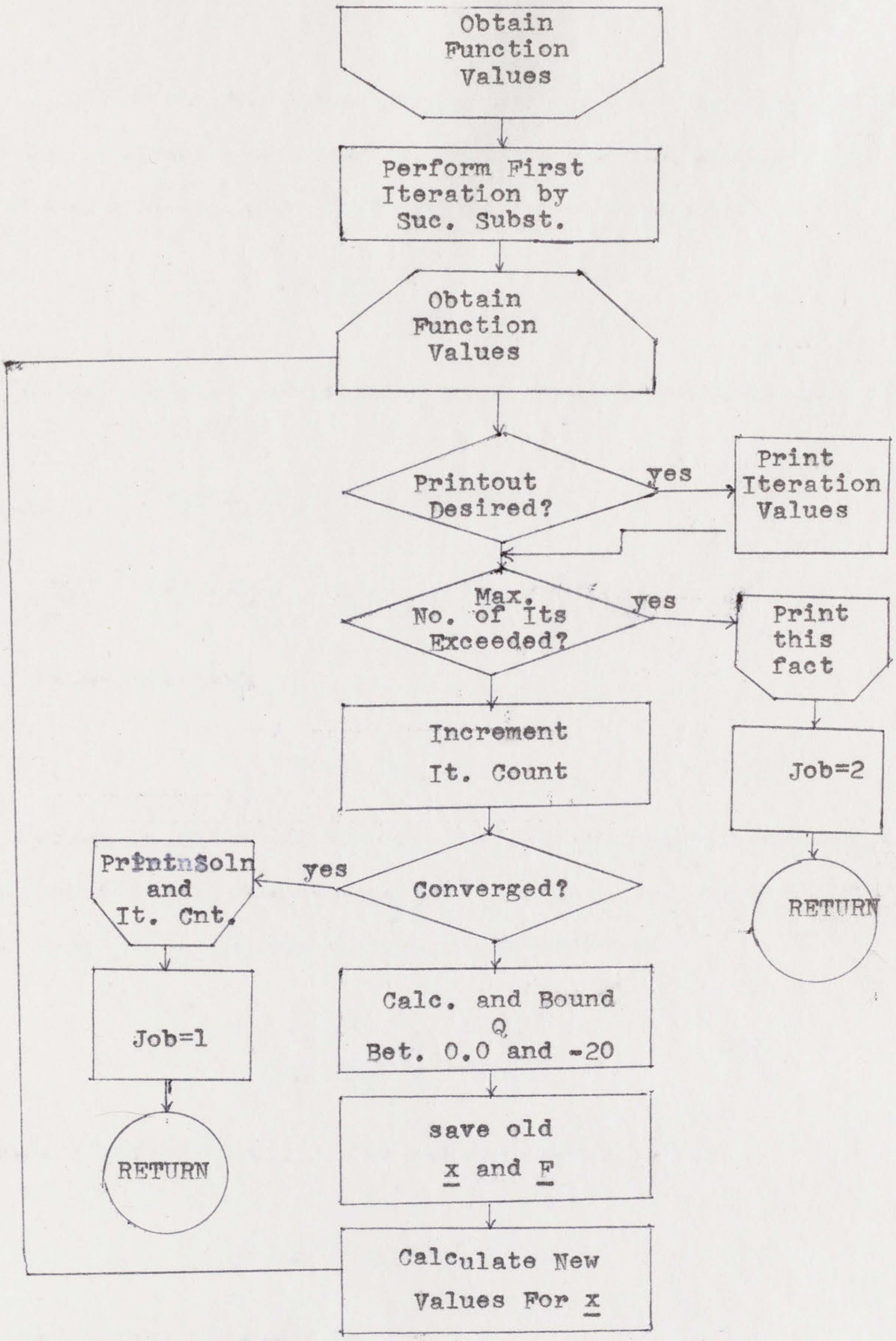


FIGURE 3 SUBROUTINE BWEG9

(iv) The Multidimensional Wegstein Method is developed along the same reasoning as the one-dimensional Wegstein Method. The recycle formulation is again stated as:

$$\underline{x}= F(\underline{x}) \tag{1}$$

A linear form is assumed for each variable in each function:

$$\begin{aligned} F_1 &= a_{11}x_1 + a_{12} + \ldots a_{1n} + b_1 \\ &\vdots \\ F_n &= a_{n1}x_1 + a_{n2} + \ldots a_{nn} + b_n \end{aligned} \tag{16}$$

or in matrix form:

$$\underline{F}= \underline{Ax}+ \underline{b} \tag{17}$$

in order to apply the method, N+1 paired values of $\underline{F}$ and $\underline{x}$ are obtained by Successive Substitution. If these are numbered i= 0,1,2,3,... N, and df and dx are defined:

$$\begin{aligned} (\underline{df})^{(i)} &= \underline{F}^{(i)} - \underline{F}^{(0)} \\ (\underline{dx})^{(i)} &= \underline{x}^{(i)} - \underline{x}^{(0)} \end{aligned} \tag{18}$$

then substitution into the equation abole yields:

$$(\underline{df}^{(i)}) = A(\underline{dx}^{(i)}) \tag{19}$$

therefore it follows that:

$$\underline{df}= \underline{Adx} \tag{20}$$

or:

$$\underline{A}= (\underline{df})(\underline{dx})^{-1} \tag{21}$$

and:

$$\underline{b}= \underline{F}- \underline{Ax} \tag{22}$$

for the next guess we want $\underline{F}= \underline{x}$:

$$\underline{x}^{(i+1)}= \underline{Ax}^{(i+1)}+ \underline{b}$$

or:

$$\underline{x}^{(i+1)}=(\underline{I}-\underline{A})^{-1}\underline{b} \tag{23}$$

If we define a convergence parameter matrix, Q, as:

$$\underline{x}^{(i+1)}= \underline{Qx}^{(i)}+ (\underline{I}-\underline{Q})\underline{F}^{(i)} \tag{24}$$

and using the values for $\underline{A}$ and $\underline{b}$ calculated above one can solve for $\underline{Q}$, as was done in the one-dimensional case. The results are summarized below:

$$\begin{aligned}
&\underline{x}^{(i+1)}= \underline{Qx}^{(i)}+ (\underline{I}-\underline{Q})\underline{F}^{(i)}\\
&\underline{Q}= (\underline{A}-\underline{I})^{-1}\underline{A}\\
&\underline{A}= (\underline{df})(\underline{dx})^{-1}
\end{aligned} \tag{25}$$

The previous N iterations are used to obtain the matrices $\underline{df}$ and $\underline{dx}$, from which the new guess $x^{(i+1)}$ is derived. The flow sheet for the method is illustrated in Fig. 4. A listing of the SUBROUTINE MVWG9 can be found in Appendix B.

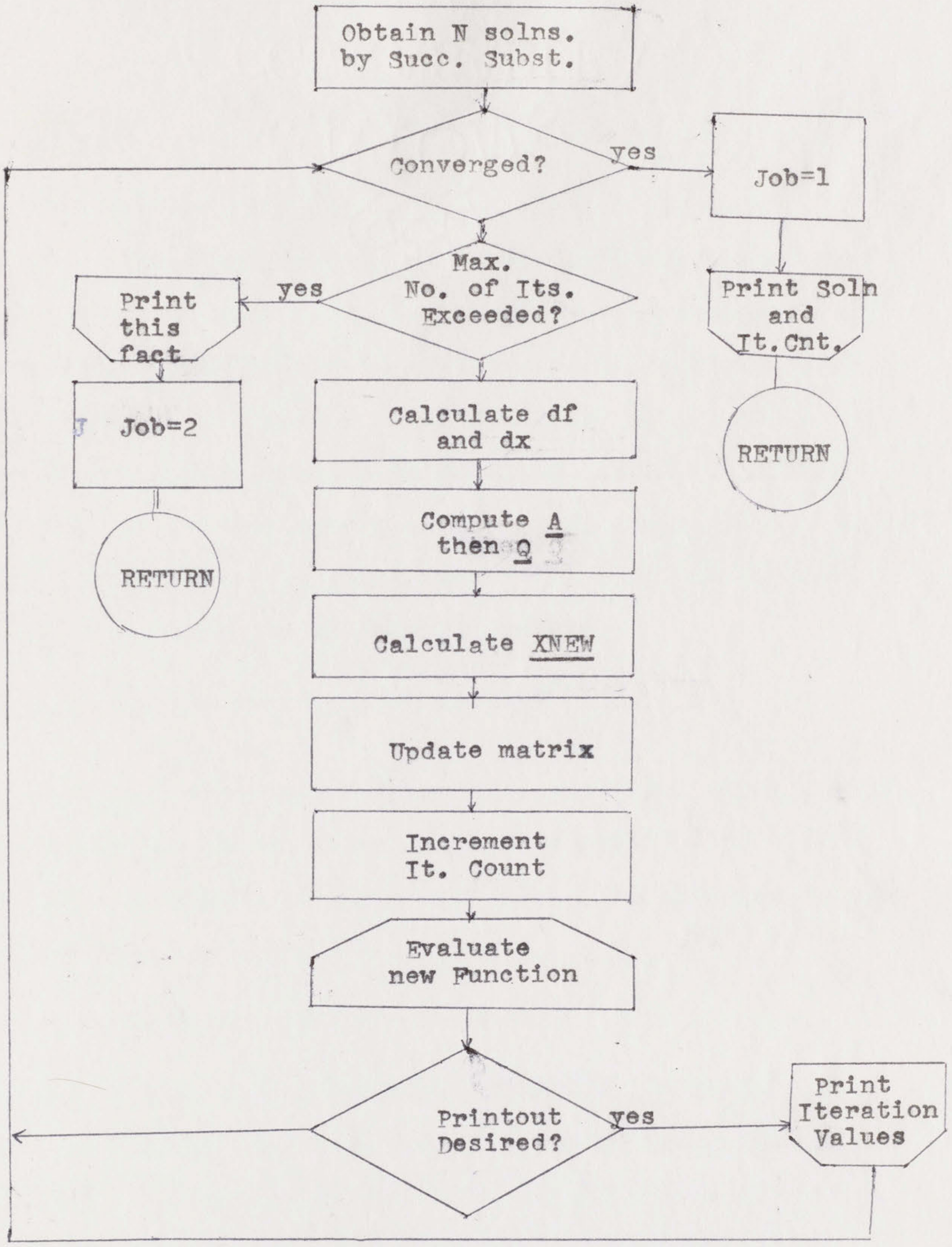


FIGURE 4 SUBROUTINE MVWG9

III ENGINEERING EXAMPLES

This section considers several engineering example problems. The first two are examples which confronted me during my course work, and the last three are examples used by previous investigators to illustrate the superiority of their iterative procedures. The executive program is written as described in the Problem Formulation, SUBROUTINE FUNV9 simulates the particular process on hand, while SUBROUTINE XQT9 initializes the recycle guess, and calls each convergence procedure to solve the simulation problem.

A: Aluminum Purification Problem[2]

This is a vapor phase deposition reaction, where crude Aluminum is fed to the first reactor, maintained at $1450^{o}K$, along with a mixture of $AlCl_3$ and $AlCl$. The reaction takes the form of:

$$AlCl_3(g) + 2Al(l) = 3AlCl(g)$$

This gas is then fed to the second reactor, which is maintained at $1350^{o}K$. The same reaction occurs here, but the equilibrium is shifted to the left, depositing purified liquid aluminum in the second reactor. (see Fig. 5).

In order to calculate the pound moles of aluminum purified per pound-mole of gas leaving Reactor 1, I chose

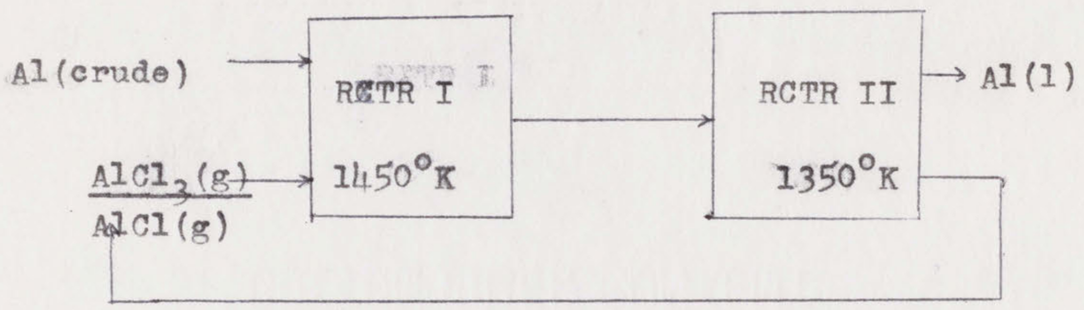


FIGURE 5 ALUMINUM PURIFICATION SET-UP

as a basis one lb.-mole of gas leaving Reactor 1. This gas will contain x_1 moles of AlCl and x_2 moles of $AlCl_3$ since the gas contains only these two components.

$$x_1 + x_2 = 1.0 \qquad (26)$$

The equilibrium constant for t his reaction is defined by the equation:

$$\ln K = (61.2 - 92{,}940/T)/R \qquad (27)$$

This yields $K_{1450} = 0.233$, $K_{1350} = 0.021$. Therefore, for Reactor 1:

$$0.233 = x_1^3 PRES_1^2/x_2^2 \qquad (27)$$

Here, $PRES_1$ is the total pressure in Reactor 1. A chlorine balance around Reactor 1 yields the amount of chlorine in the product stream, x_3:

$$x_3 = x_1 + 3x_2 \qquad (28)$$

This also determines the chlorine balance for the second reactor, for if there are x_4 moles of AlCl in the product stream, there must be $(x_3 - x_4)/3$ moles of $AlCl_3$ in the stream (this assumes the chlorine is insoluble in the liquid aluminum). Substituting these relations into the equilibrium relation yields, for the second reactor;

$$9x_4^3 PRES_2^2/((x_3 - x_4)(x_3 + 2x_4)) = 0.021 \qquad (29)$$

x_5, the amount of aluminum formed per lb-mole of gas leaving Reactor 1, can now be determined by using an aluminum balance around Reactor 2.

$$x_5 = x_1 + x_2 - x_4 - (x_3 - x_4)/3 \qquad (30)$$

Having set the temperatures and pressures as in the problem statement, we can now use an iterative procedure to determine the values of all these unknowns. (see Fig. 6). The number of iterations required by the four procedures being studied were obtained, using the stated pressures and various other pressures for the first reactor. (see Figs. 7-10). The Bounded Wegstein method failed to approach the solution in one hundred iterations for a pressure of one atmosphere in the first reactor, even though it obtained solutions for all the higher pressure simulations.

The Multidimensional Wegstein method obtained solutions much more quickly then did any of the other methods, with Successive Substitution close behind in efficiency. The Newton-Raphson method is much slower because of the number of function evaluations necessary to generate the derivative matrix for each iteration. Obviously, the trade-off here on which method is best would be influenced greatly by the length of time required to obtain each function evaluation, such that any one of the four methods might be the best. The differences become even less pronounced as the pressure in

```
// FOR
      SUBROUTINE FUNV9(X,F,JOBB)
C
C*****  THIS SUBROUTINE SOLVES AN ALUM. PURIF. PROBLEM
C
      DIMENSION  X(10), F(10)
      GO TO(100,150,200,300,400),JOBB
C****  THIS SECTION SOLVES X=F(X)
100   CONTINUE
      F(1)= (.233*x(2)/(PRES1**2))**0.333
      F(2)= 1.0-x(1)
      F(3)= x(1)+ 3.0*x(2)
      F(4)= (0.233E_02*(x(3)-x(4))*(x(3)+2.*x(4))/PRES2**2
     1 )**0.333
      F(5)= X(1)+ X(2)- X(4)-(X(3)- X(4))/3.0
      RETURN
C****  THIS SECTION SOLVES FOR F(X)= 0
150   CONTINUE
      F(1)= .233-PRES1**2*X(1)**3/X(2)
      F(2)= 1.0-X(1)- X(2)
      F(3)= X(3)- A(1)- 3.0*X(2)
      F(4)= .021- 9.*X(4)**3*PRES2**2/((X(3)-X(4))*(X(3)
     1 + 2.0*X(4)))
      F(5)= X(5)-X(1)-X(2)+X(4)+ (X(3)_X(4))/3.0
      ICNT= ICNT + 1
      RETURN
C****  THIS SECTION SETS INITIAL CONDITIONS
200   CONTINUE
      PRES1= 1.0
      PRES2= 1.0
      ICNT= 0
      N= 5
      RETURN
C****  THIS SECTION FOR ALTERING CONDITIONS
300   CONTINUE
      ICNT= 0
      PRES1= PRES1+ 2.0
      IF(PRES1- 5.0) 325.325.350
325   WRITE(3.9000) PRES1,PRES2
9000  FORMAT('1 ALUM. PURIF. PROB., PRES1=',F10.2,' PRES2=',F10.2)
      RETURN
350   CONTINUE
      IF(PRES1- 10.0) 360,370,370
360   PRES1= 10.0
      GO TO 325
```

(Continued)

```
370     PRES1= PRES1+ 5.0
        IF(PRES1- 40.0) 325,325,380
380     CONTINUE
        CALL EXIT
C****   THIS SECTION PRINTS NO. OF ITS. REQUIRED FOR SOLN.
400     CONTINUE
        WRITE(3,9100) ICNT
9100    FORMAT(' SOLN REQUIRED',I6,' FUNCTION EVALS')
        ICNT= 0
        RETURN
        END
```

FIGURE 6 ALUMINUM PURIFICATION LISTING

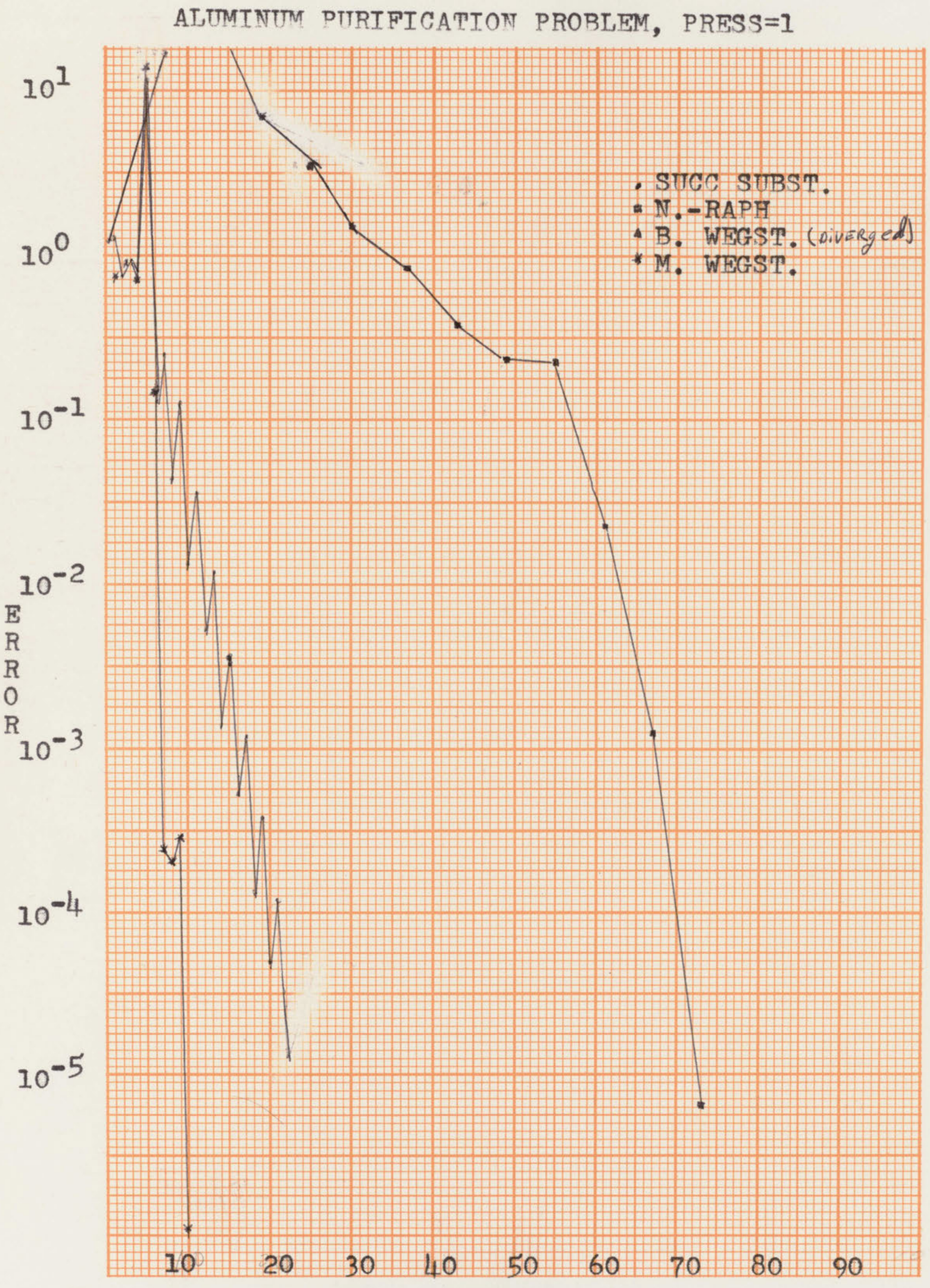

ALUMINUM PURIFICATION PROBLEM, PRESS=1
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
ERROR
10
20
30
40
50
60
70
80
90
FUNCTION EVALUATIONS
• SUCC SUBST.
▪ N.-RAPH
▴ B. WEGST. (diverged)
* M. WEGST.

Figure 7

ALUMINUM PURIFICATION PROBLEM, PRESS=3

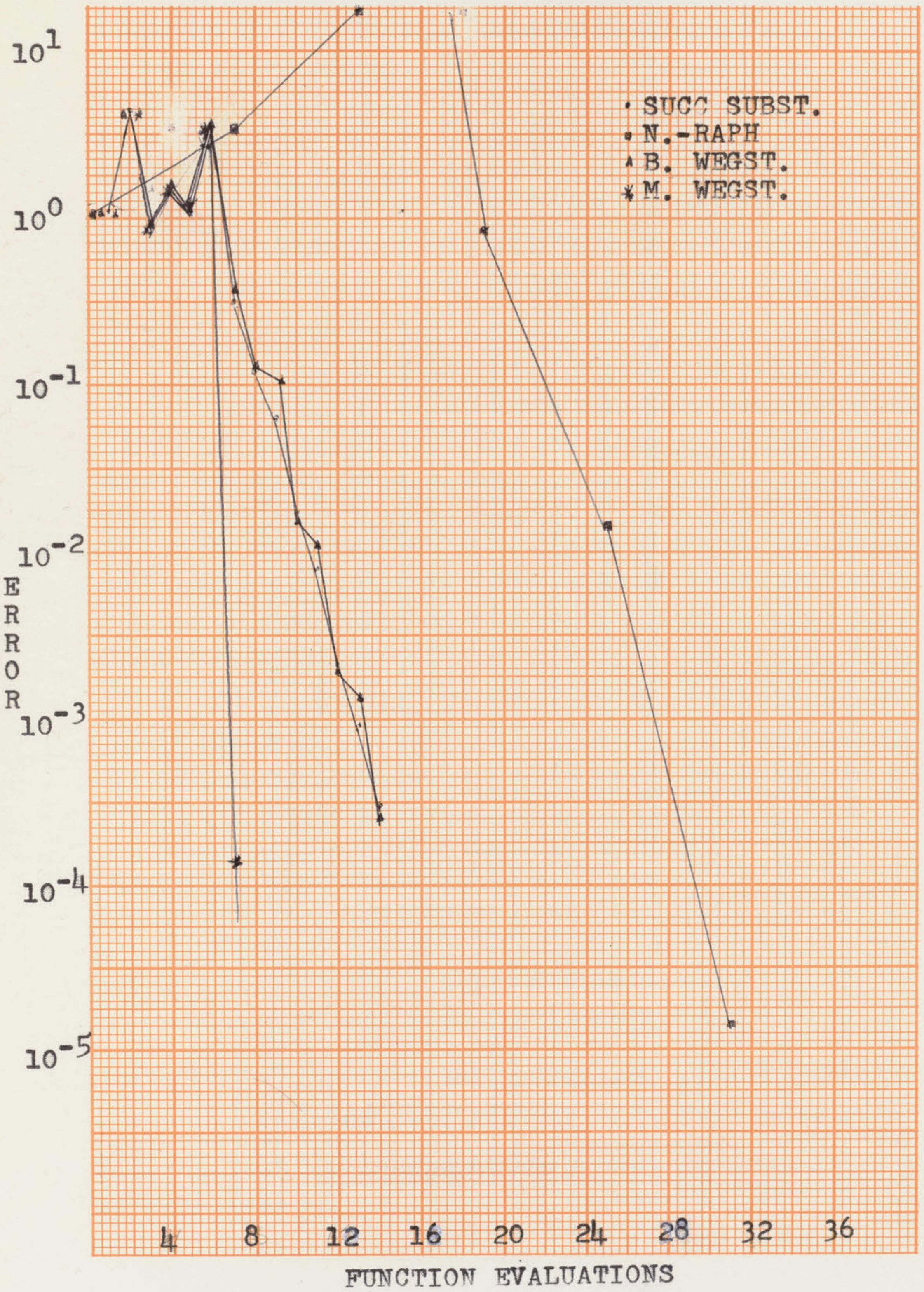


Figure 8

ALUMINUM PURIFICATION PROBLEM, PRESS= 5

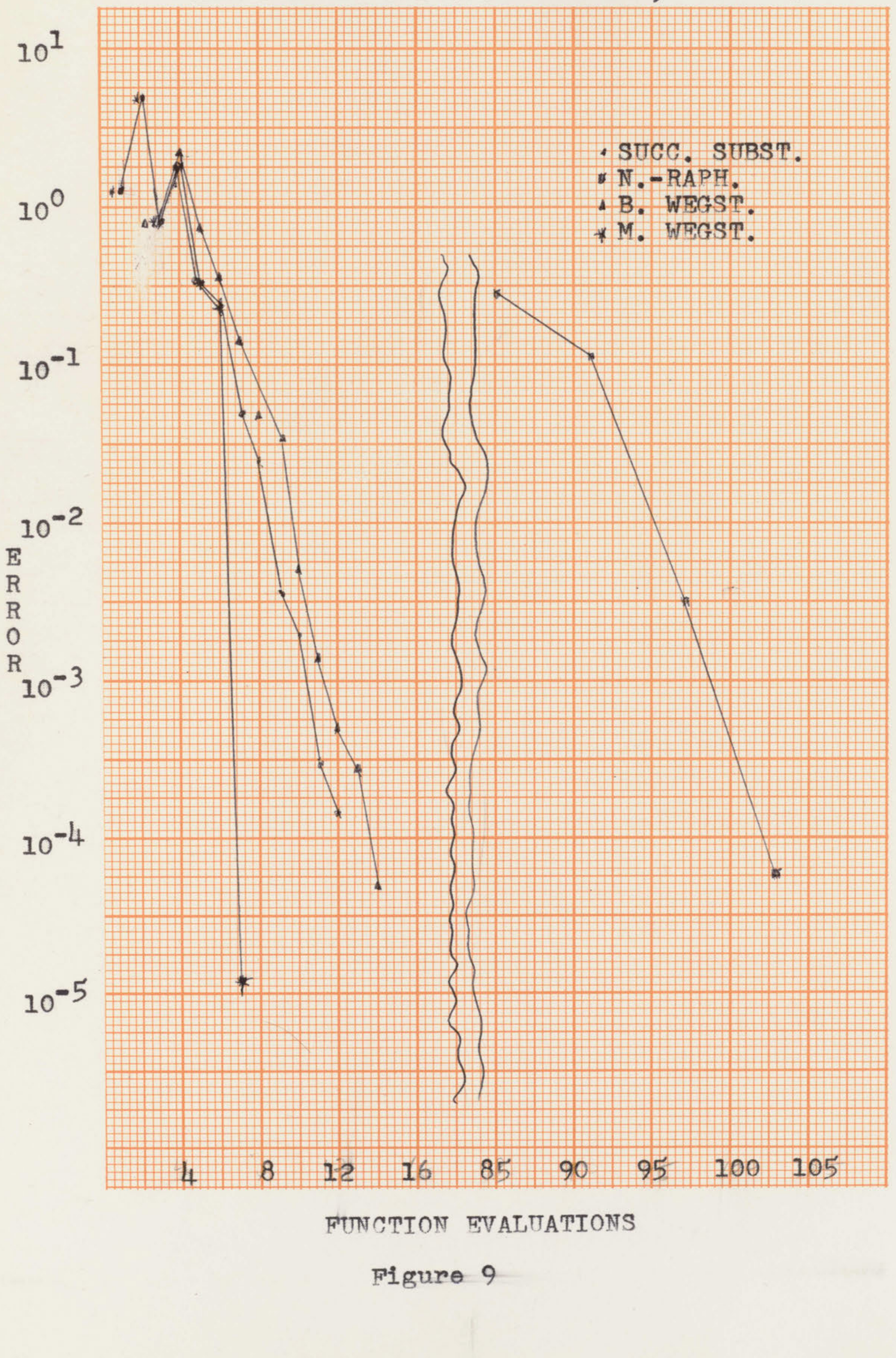
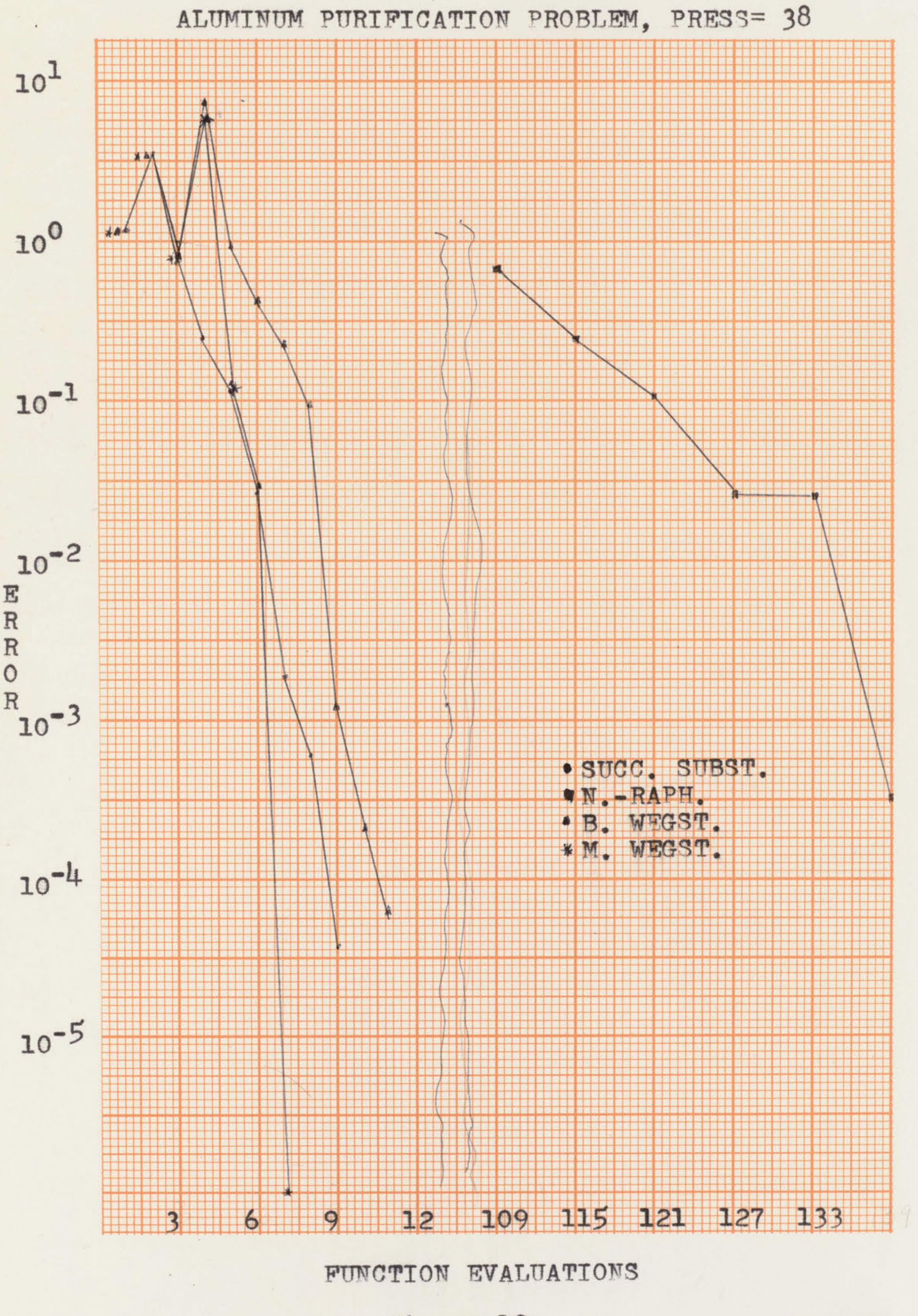


Figure 9



ALUMINUM PURIFICATION PROBLEM, PRESS= 38
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
ERROR
3
6
9
12
109
115
121
127
133
• SUCC. SUBST.
■ N.-RAPH.
▲ B. WEGST.
* M. WEGST.
FUNCTION EVALUATIONS

Figure 10

the first reactor is increased.

B: Combustion Problem[3]

The second example considers an important industrial reaction, the Water-Gas shift reaction. The problem statement is as follows:

A fuel oil analyzing 86% C., 11% H, and 3% Inerts is burned with 50% of the theoretical air needed for complete combustion. The gas mixture is brought to equilibrium at 1800°F. What is the compostion of the product gas if these two reactions are the only important ones?

$$\text{I.}\quad CO_2 + H_2 = CO + H_2O \qquad K_{p,1800} = 1.66$$

$$\text{II.}\quad CO + 3H_2 = CH_4 + H_2O \qquad K_{p,1800} = 1.48\times10^{-4}$$

ans:

Chose as a basis, 100 lbs. oil.

		moles present	moles O_2 needed
C	86%	7.17	7.17
H	11%	11.0	2.75
I	3%	0.00	0.00
			9.92

one half the theoretical O_2 need is 4.96 moles, so 18.65 moles of N_2 are introduced with the air. Defining the amounts of each species present as follows: CO, a; CO_2, b; H_2O, c; CH_4, d; and H_2, e, one can obtain the following equations to simulate the system:

C balance	$7/17= a+ b+ d$
H_2 balance	$5.50= c+ 2d+ e$
O_2 balance	$4.96= a/2+ b+ c/2$
Equil. Rctr. 1	$1.66= ac/(be)$
Equil. Rctr. 2	$1.48\mathrm{x}10^{-4}= dc(18.65+a+b+c+d+e)^2/(ae^3\mathrm{PRESS}^2)$

Where PRES is the pressure operated at. SUBROUTINE FUNV9 was formulated to simulate this problem (see Fig. 11). Tests were run for the various convergence procedures with the pressure varying from one to thirty-two atmospheres. The speed of solving this problem appears to be independent of the pressure, for all the test runs proceeded at the same pace (see Figs. 12-13). The Multidimensional Wegstein method proved much quicker than the method of Successive Substitution. The Bounded Wegstein and the Newton-Raphson methods both failed to obtain solutions to the problem.

C: Ethylene Dichloride[4]

Napthali proposed a hypothetical process for the making of ethylene dichloride, where a recycle stream from the separator is mixed with the feed-stock and fed into a mixed reactor, which converts 90% of the ethylene per pass. Its product

```
//FOR
        SUBROUTINE FUNV9(X,F,JOBB)
        GO TO(100,200,300,400,500),JOBB
C****   THIS SECTION SOLVES F(X)=X
100     CONTINUE
        F(1)= 7.17-X(2)-X(4)
        F(2)= 4.96-(X(1)+X(3))/2.0
        F(3)= 1.66*X(2)*X(5)/X(1)
        PART= 0.0
        DO 125 I= 1,5
 125    PART= PART+ X(I)
        PART= ((PART+ 18.65)**2)*X(3)
        F(4)= .148E-05*PRES**2*X(1)*X(5)**3/PART
        F(5)= 5.5-X(3)-2.0*X(4)
        ICNT= ICNT+ 1
        RETURN
C****   THIS SECTION SOLVES F(X)= 0
200     CONTINUE
        F(1)= 7.17-X(1)-X(2)-X(4)
        F(2)= 4.96-X(2)-(X(1)+ X(3))/2.0
        F(3)= 1.66-X(1)*X(3)/(X(2)*X(5))
        F(5)= 5.5-X(3)-X(5)-2.0*X(4)
        PART= 0.0
        DO 225 I=1,5
225     PART= PART+ X(I)
        PART= (PART+ 18.65)**2
        F(4)= X(1)*X(5)**3*PRES**2
        F(4)= .148E-05- X(4)*X(3)*PART/F(4)
        ICNT= ICNT+ 1
        RETURN
C****   THIS SECTION SETS THE INITIAL CONDITIONS
300     CONTINUE
        PRES= 1.0
        ICNT= 0
        N= 5
        GO TO 450
C****   THIS SECTION FOR CHANGING INITIAL VALUES OF PARAMS.
400     CONTINUE
        ICNT= 0
        PRES= PRES*2.0
        IF(PRES-50.0) 450,425,425
```

(Continued)

COMBUSTION PROBLEM   PRESS= 1.0

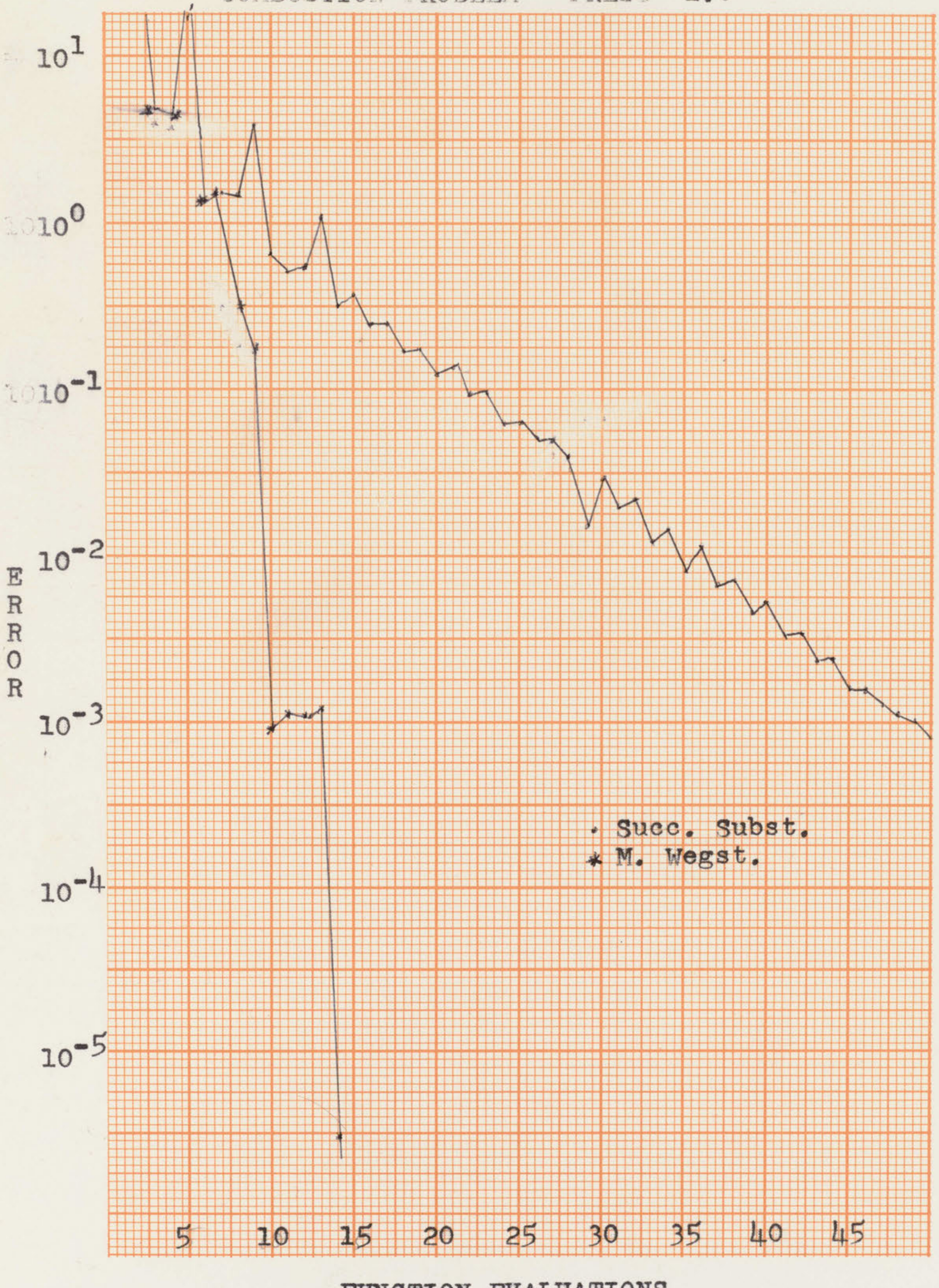


Figure 12

COMBUSTION PROBLEM PRESS= 32.0

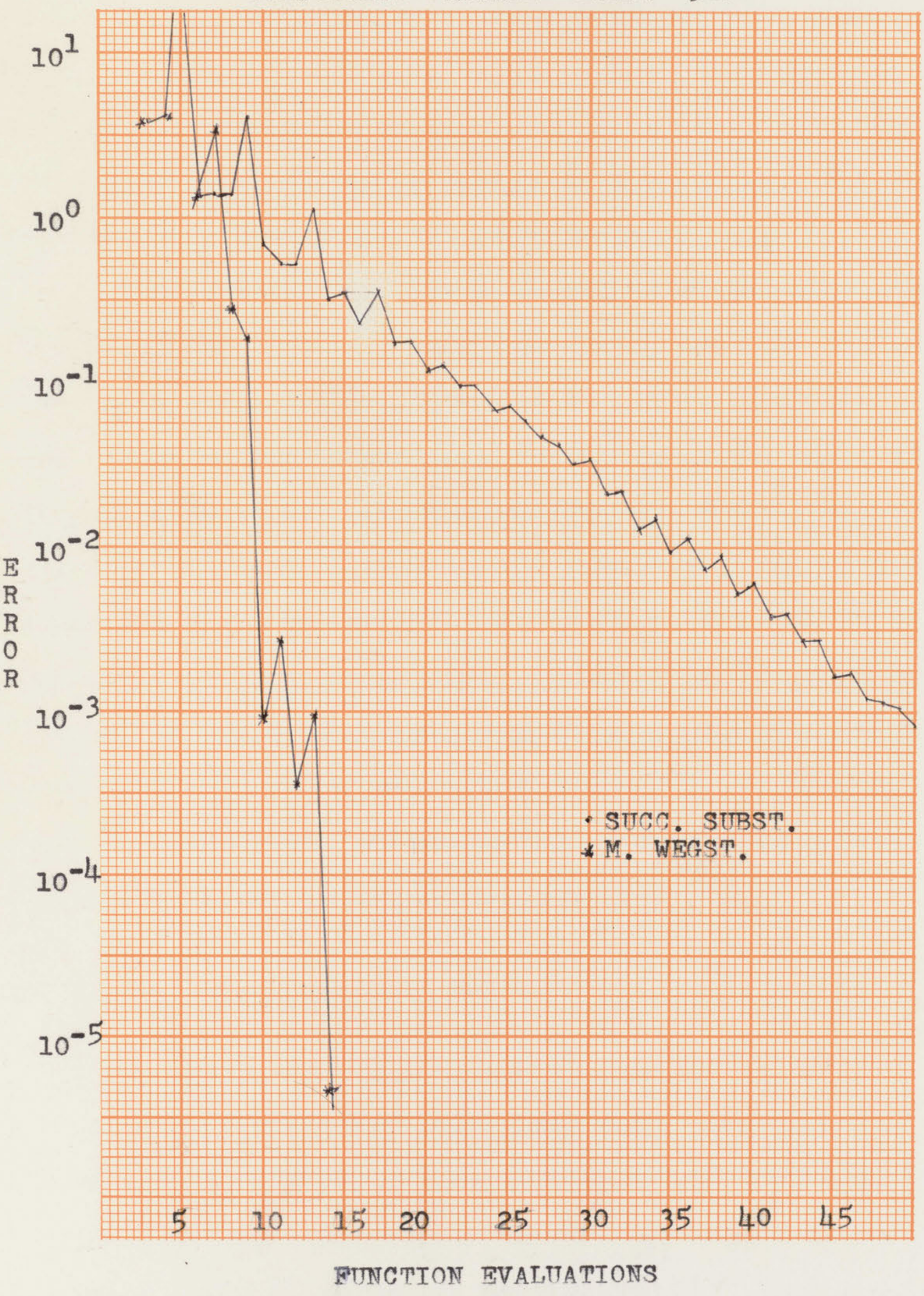


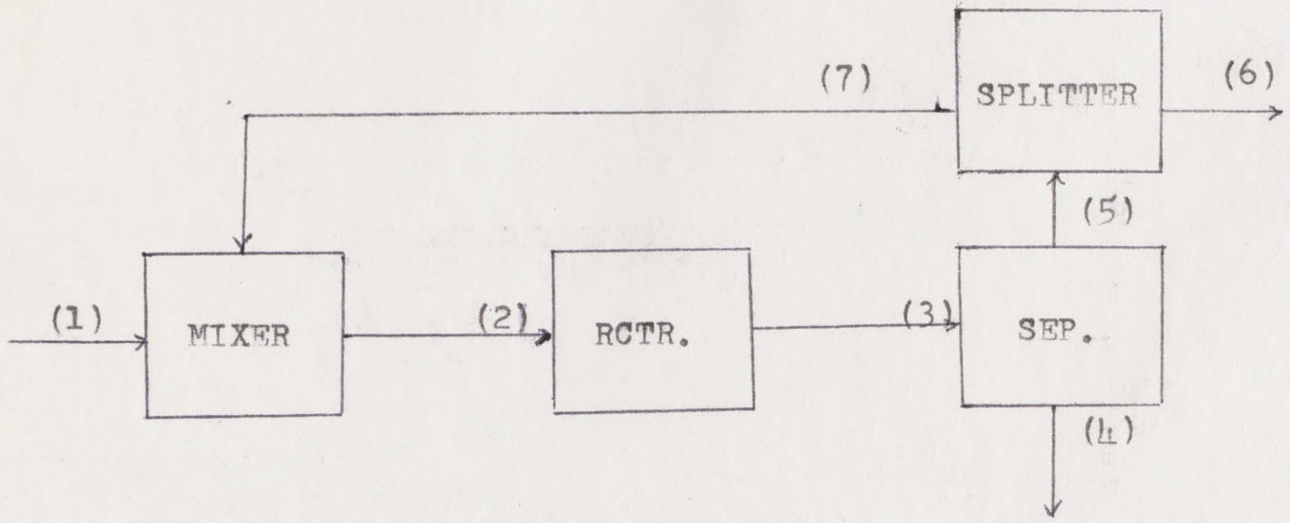
Figure 13

stream is fed to a separator whose overhead will contain 98% of the chlorine entering the unit, 92% of the ethylene, and 0.1% of the ethylene dichloride entering the unit. Five percent of the overhead is purged by a splitter. For the mathematical formulation of the problem see Fig. 14.

SUBROUTINE FUNV9 was developed to effect this simulation (see Fig. 15). Two test runs were made, one with the feed containing 45% ethylene, 45% chlorine, and 10% inerts; the other with the feed containing 50% chlorine and 50% ethylene. For both simulations the Bounded Wegstein Method was the quickest, followed closely by the Multidimensional Wegstein Method(see Figs. 16 and 17). Successive Substitution is very slow, so that if any reasonable degree of accuracy is required, this method would be an unlikely choice. The Bounded Wegstein method would be the overwhelming choice, it is quicker than the Multidimensional method, both in the speed of its iterations and in the amount of iterations required.

D: Photochemical Reaction Problem^5

Kwon proposed a photochemical reaction problem. The equations he derives to model this can be seen around statement 100 of the listing for SUBROUTINE FUNV9 (see Fig. 18), the routine developed to simulate this situation. The reaction constants and flow rates of the various species follow statement



Strm.#	1	2	3	4	5	6	7
C_2H_4	A	$A+x_{17}$	$.1x_{12}$	$.08x_{13}$	$.92x_{31}$	$.05x_{51}$	$.95x_{51}$
Cl_2	B	$B+x_{27}$	$x_{22}-.9x_{12}$	$.02x_{23}$	$.98x_{32}$	$.05x_{52}$	$.95x_{52}$
$C_2H_4Cl_2$	—	x_{37}	$x_{32}+.9x_{12}$	$.999x_{33}$	$.001x_{33}$	$.05x_{53}$	$.95x_{53}$
I	I	$I+x_{47}$	$I+x_{47}$	—	x_{43}	$.05x_{57}$	$.95x_{54}$

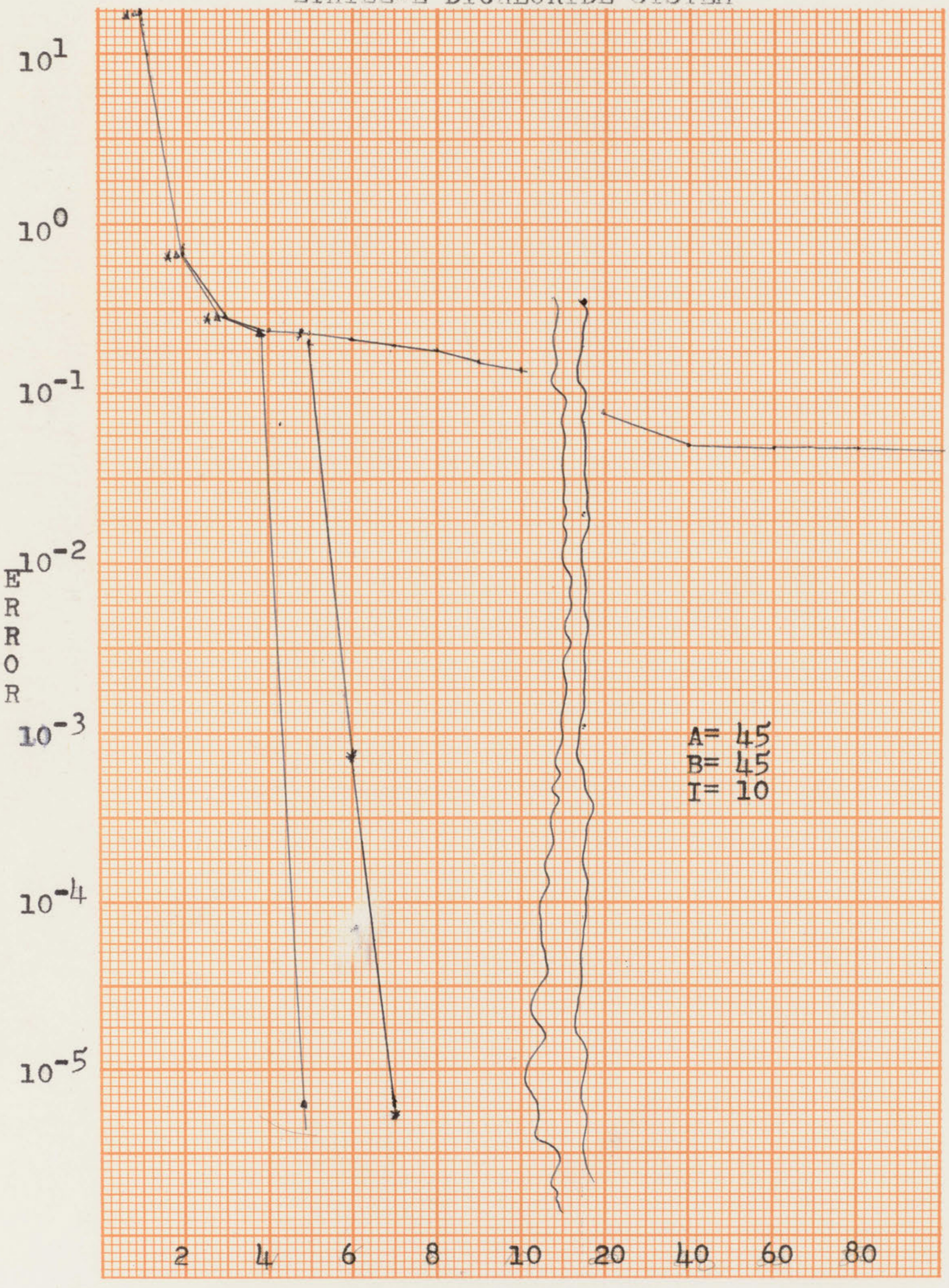
where x_{ij} is the amount of component i in stream no. j. where i=1, is C_2H_4, i=2, is Cl_2, i=3 is $C_2H_4Cl_2$, i=4 is inerts.

Figure 14 ETHYLENE DICHLORIDE SYSTEM

```
// FOR
        SUBROUTINE FUNV9(X,F,JOBB)
        DIMENSION X(10),F(1)
C*****  THIS SUBROUTINE SOLVES ETHYLENE DICHLORIDE SYSTEM
        GO TO(100,100,300,400,5000,JOBB
C*****  SOLVES FUNCT. FOR X=F(X)
100     CONTINUE
        F(1)= 0.0874*(A+ X(1))
        F(2)= 0.931*(B+X(2)-0.9*(A+ X(1))
        F(3)= 0.00095*(X(3)+ .9*(A+X(1)))
        F(4)= .95*(XI+ X(4))
        ICNT=ICNT+ 1
        IF(JOBB-2) 200,150,200
 C----  THIS SECTION SOLVES FOR F(X)= 0
150     CONTINUE
        DO 175 I= 1,4
175     F(I)= X(I)-F(I)
200     CONTINUE
        RETURN
C****   THIS SECTION INITIALIZES PARAMETERS
300     CONTINUE
        A= 45.0
        B= 45.0
        XI= 10.0
        ICNT= 0
350     WRITE(3,9000) A,B,XI
9000    FORMAT('1 ETHYLENE DICHLORIDE PROBLEM, A=',F12.2,'
     1  B=',F12.2,' C=',F12.2)
        RETURN
C****   INITIALIZE SECOND PROBLEM
400     CONTINUE
        IF(B-47.0) 410,450,450
410     ICNT= 0
        A= 50.0
        B= 50.0
        XI= 0.0
        GO TO 350
450     CONTINUE
        CALL EXIT
C****   RESET ICNT, PRINT RESULTS
500     WRITE(3,9100) ICNT
 9100   FORMAT(' SOLUTION REQRD',I6,' FUNCTION EVALUATIONS')
        ICNT= 0
        RETURN
        END
```

Figure 15 ETHYLENE DICHLORIDE LISTING

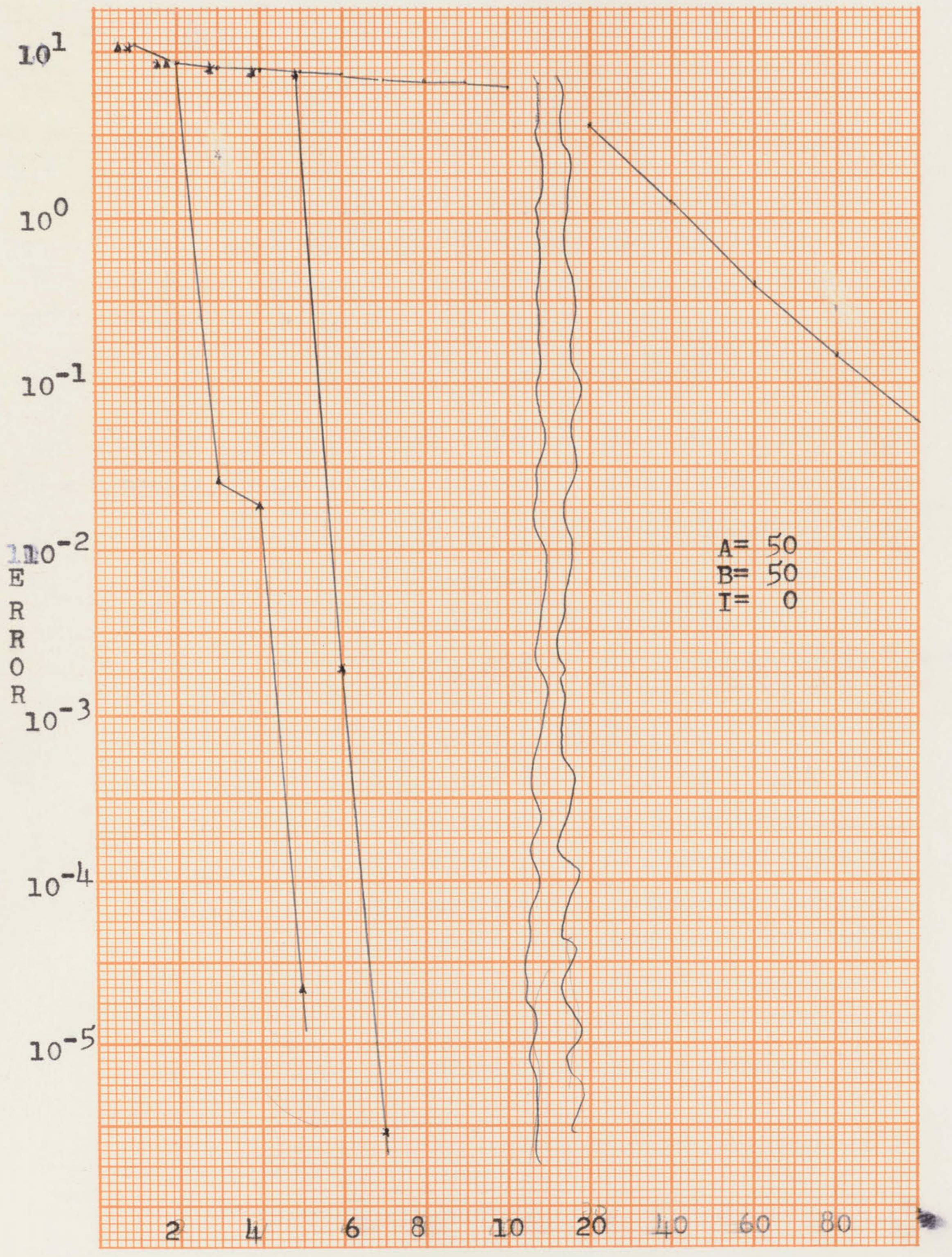
ETHYLENE DICHLORIDE SYSTEM



FUNCTION EVALUATIONS

Figure 16

ETHYLENE DICHLORIDE SYSTEM



FUNCTION EVALUATIONS

Figure 17

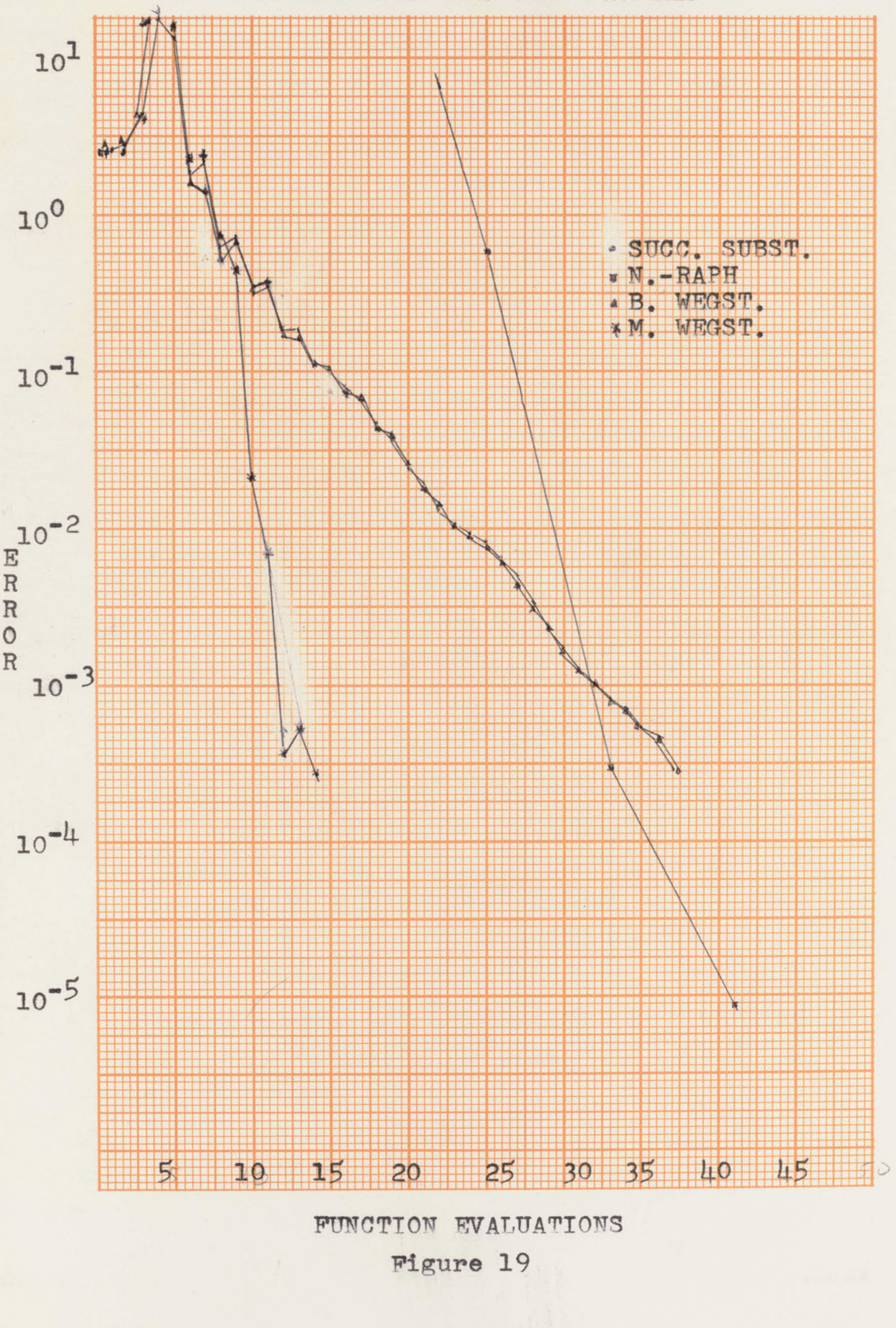
```
// FOR
      SUBROUTINE FUNV9(X,F,JOBB)
      DIMENSION X(10),F(10)
C****   SOLVES THE PHOTOCHEMICAL PROBLEM OF KWON
      GO TO(100,150,200,300,500),JOBB
C****   SOLVES FOR PROBLEMS OF FORM X=F(X)
100   CONTINUE
      ICNT= ICNT+ 1
      F(1)= F1/(FF+RK1*X(1)+RK4*X(6)*(RLITE**0.5))
      F(2)= F2/(FF+RK1*X(1)+2.0*RK2*X(3))
      F(3)= F3/(FF+ RK2*X(2))
      F(4)= 2.0*RK1*X(1)*X(2)/(FF+RK3*X(5))
      F(5)= 2.0*RK2*X(2)*X(3)/(FF+RK3*X(4))
      F(6)= 2.0*RK3*X(4)*X(5)/(FF*RK4*X(1)*RLITE**0.5)
      F(7)= 2.0*RK4*X(1)*X(6)*RLITE**0.5/FF
      RETURN
C****   SOLVES FOR PROBLEMS OF FORM F(X)= 0
150   CONTINUE
      ICNT = ICNT+ 1
      F(1)= F1-FF*X(1)-RK1*X(1)*X(2)-RK4*X(1)*X(6)*RLITE**0.5
      F(2)= F2-FF*X(2)-RK1*X(1)*X(2)-2.0*RK2*X(2)*X(3)
      F(3)= F3-FF*X(3)-RK2*X(2)*X(3)
      F(4)= 2.0*RK1*X(1)*X(2)-FF*X(4)-RK3*X(4)*X(5)
      F(5)= 3.0*RK2*X(2)*X(3)_FF*X(5)-RK4*X(4)*X(5)
      F(6)= 2.0*RK3*X(4)*5(5)-FF*X(6)-RK4*X(1)*X(6)*RLITE**0.5
      F(7)= 2.0*RK4*X(1)*X(6)*RLITE**0.5-FF*X(7)
      RETURN
C****   INITIALIZE GUESS AND PARAMETERS
200   CONTINUE
      ICNT= 0
      DO 350 I=1,7
350   X(I)= 0.0
      WRITE(3,9000)
9000  FORMAT(' SOLVES PHOTOCHEMICAL PROBLEM')
      RK1= 17.6
      RK2= 73.0
      RK3= 51.3
      RK4=23.0
      F1= 3.0
      F2= 4.75
      F3= 1.25
```

(Continued)

```
        FF= 9.0
        RLITE= 0.6
        RETURN
C****   THIS SECTION TERMINATES THE RUN
300     CONTINUE
        CALL EXIT
C****   THIS SECTION PRINTS ITCNT. AND ZERO'S IT
500     CONTINUE
        WRITE(3,9100) ICNT
9100    FORMAT(' SOLUTION REQUIRED',I6,' FUNCTION EVALUATIONS')
        ICNT=0
        RETURN
        END
```

Figure 18 PHOTOCHEMICAL PROBLEM LISTING

PHOTOCHEMICAL REACTION PROBLEM



FUNCTION EVALUATIONS

Figure 19

number 350. The problem was solved by each of the four methods (see Fig. 19). The Multidimensional Wegstein Method was much better than any of the other methods at solving this problem.

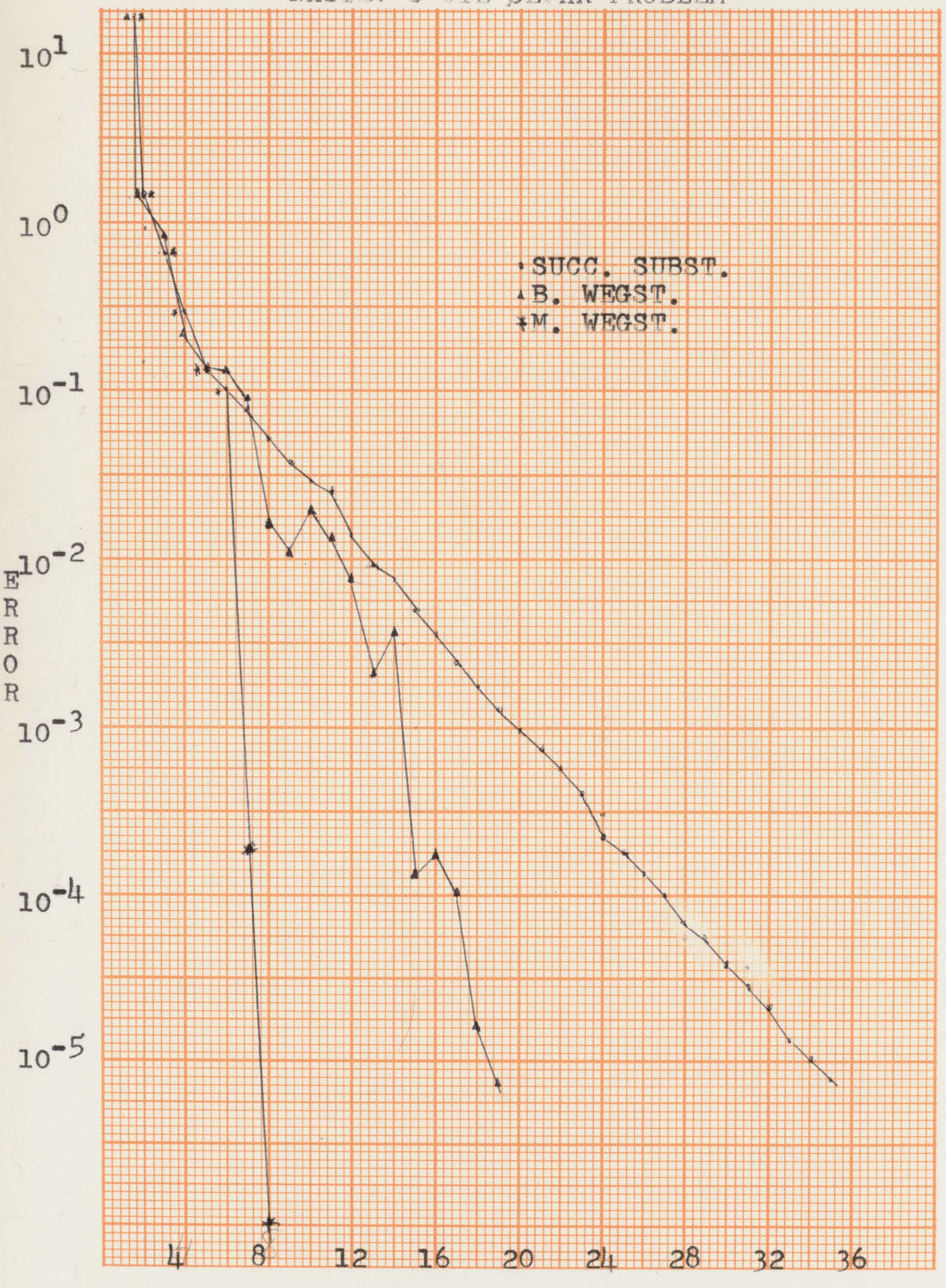
E: Oil Separation Problem[6]

Nagiev proposed an oil separation problem, described by the equations following statement number 100 of the listing for SUBROUTINE FUNV9, which was developed to simulate this problem. (see Fig. 20). The results of the simulation are plotted in Fig. 21. The Multidimensional Wegstein Method again was much better than the other methods. The Newton-Raphson method failed in an attempt to invert a matrix, so it would require higher precisions to solve the problem.

```
// FOR
        SUBROUTINE FUNV9(X,F,JOBB)
        DIMENSION X(10),F(10)
C****   THIS SUB SOLVES NAGIEV'S OIL SEP PROB.
        GO TO(100,100,200,300,500),JOBB
C****   SOLVES FOR FUNCT SOLVED BY X=F(X)
100     CONTINUE
        ICNT= ICNT+ 1
        F(1)= 1000.+ .4624*X(1)+ .0436*X(2)
        F(2)= 200.+ .235*X(1)+ .67*X(2)+ .1667*X(5)+.05*X(7)
        F(3)= 100.+ .008*X(1)+.061*X(2)+;445*X(3)+;001*X(4)
        F(4)= 200+0.021*X(1)+.0022*X(2)+.268*X(4)+.011*X(6)
        F(5)= 50.+ 0.0032*X(1)+ .0025*X(2)+ .213*X(3) +
     1  .0833*X(5)+ .05*X(7)
        F(6)= 70.0+ .0017*X(1)+ .0014*X(2)+ .29*X(4)+ .482*X(6)
        F(7)= (.75*X(5)+ .08*X(7))/.27
        IF(JOBB-2) 190,150,190
C****   THIS SECTION SOLVES FOR F(X)= 0
150     CONTINUE
        DO 175 I=1,7
175     F(I)= X(I)-F(I)
190     CONTINUE
        RETURN
C****   SECTION INITIALIZES ALL THE PARAMETERS
200     CONTINUE
        ICNT= 0
        DO 350 I= 1,7
350     X(I)= 0.0
        WRITE(3,9000)
9000    FORMAT(' SOLVES THE NAGIEV EXAMPLE PROBLEM')
        RETURN
300     CONTINUE
        CALL EXIT
C****   PRINT ITCNT AND REZERO IT
500     CONTINUE
        ICNT=0
        RETURN
        END
```

Figure 20 NAGIEV'S OIL SEPAR PROBLEM

NAGIEV'S OIL SEPAR PROBLEM



FUNCTION EVALUATIONS

Figure 21

IV. MATHEMATICAL EXAMPLES

A: Linear Variables, Linear Interaction.

The purpose of these examples is to determine if it is possible to make relevent generalizations as to when one iterative method will perform better than another in solving certain recycle problems. Since, in the vicinity of the solution, all problems appear linear, I investigated this case first. The simplest equations of this form is the set:

$$\begin{aligned} x_1 &= Ax_1 + Bx_2 + C \\ x_2 &= Ax_2 + Bx_1 + C \end{aligned} \qquad (1)$$

By setting $A=0.5$, and $C=5.0$, and varying the value of B, we can measure the sensitivity of solution procedures to the amount of interactions amongst the variables. Thus, for $B=0.0$, there is no interaction between the variables, and as the magnitude of B increases, the degree of interaction also increases.

The solution to this problem is: $x_1 = x_2 = C/(1-A-B)$, while the eigenvalues of the system of equations are: $\lambda = A \pm B$. Successive Substitution only obtains solutions for $|\lambda| \leq 1.0$, so that for $B \geq 0.5$, it will diverge.

In order to obtain a clearer idea of how the other methods were affected by the value of B, however, I did a panoramic "spot-check" of the value B. I did this by solving the problem using various values of B, originally spanning several orders of magnitude, by each of the methods and noting how many iterations are required to achieve a specified tolerance(.01), see fig. 22. Since both Successive Substitution and the Bounded Wegstein's methods ran into trouble near B=1.0, I expanded the search in this area, see fig. 23. In order to explore the area where the Multidimensional Wegstein's and the Newton-Raphson Method have problems solving the problem, I expanded the search to include values of B between 100 and 10,000, see fig. 24.

Based on these exploratory runs, I chose the following values of B for more detailed review: 0.0, 0.01, 0.1, 0.2, 0.4, 1.0, 200.0, 800.0. These values of B correspond to regions where one of the methods of solution required substantially more iterations than it previously needed. The results of these runs are plotted in figs. 25-31.

For all values of B, Successive Substitution tends to level out, so that if increased precision is required, the method becomes much slower.

The Bounded Wegstein's Method is the best method for the case of no interaction amongst the variables, for here it proceeds directly to the solution in one iteration. As inter-

B	0.0	10^{-4}	10^{-2}	1.01	10^{2}	10^{4}	10^{6}
SUC. SUB	7	7	7	#	#	#	#
B. WEG	3	3	3	#	#	#	#
N.-RAPH	4	4	4	¢	7	4	7
M. WEG	6	4	5	7	6	#	#

¢-matrix inversion failed
#-method diverged

figure 22 "spot check"

B	0.1	0.2	0.3	0.4	0.7	0.9	1.0
SUC.SUB.	9	11	15	24	#	#	#
B. WEG	3	5	6	10	#	#	#
N.-RAPH	4	4	4	4	4	7	4
M. WEG	3	3	3	3	3	3	3

#- method diverged

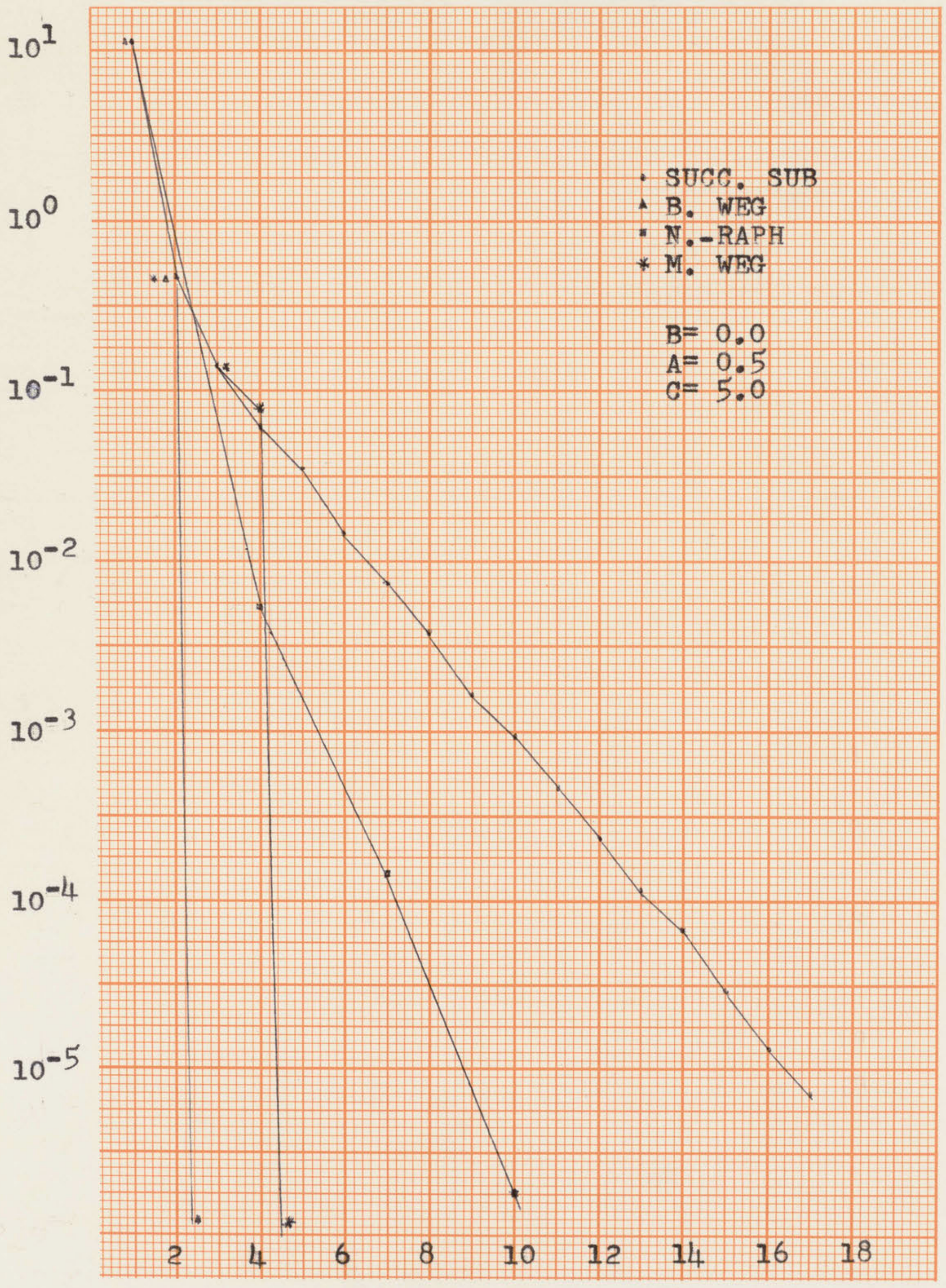
figure 23 "spot check"

B	100	200	300	500	800	900	1000	3000	5000
M.WEG	5	7	7	10	12	14	19	#	#
N.RAPH	7	7	7	7	7	7	7	7	7

#-method diverged

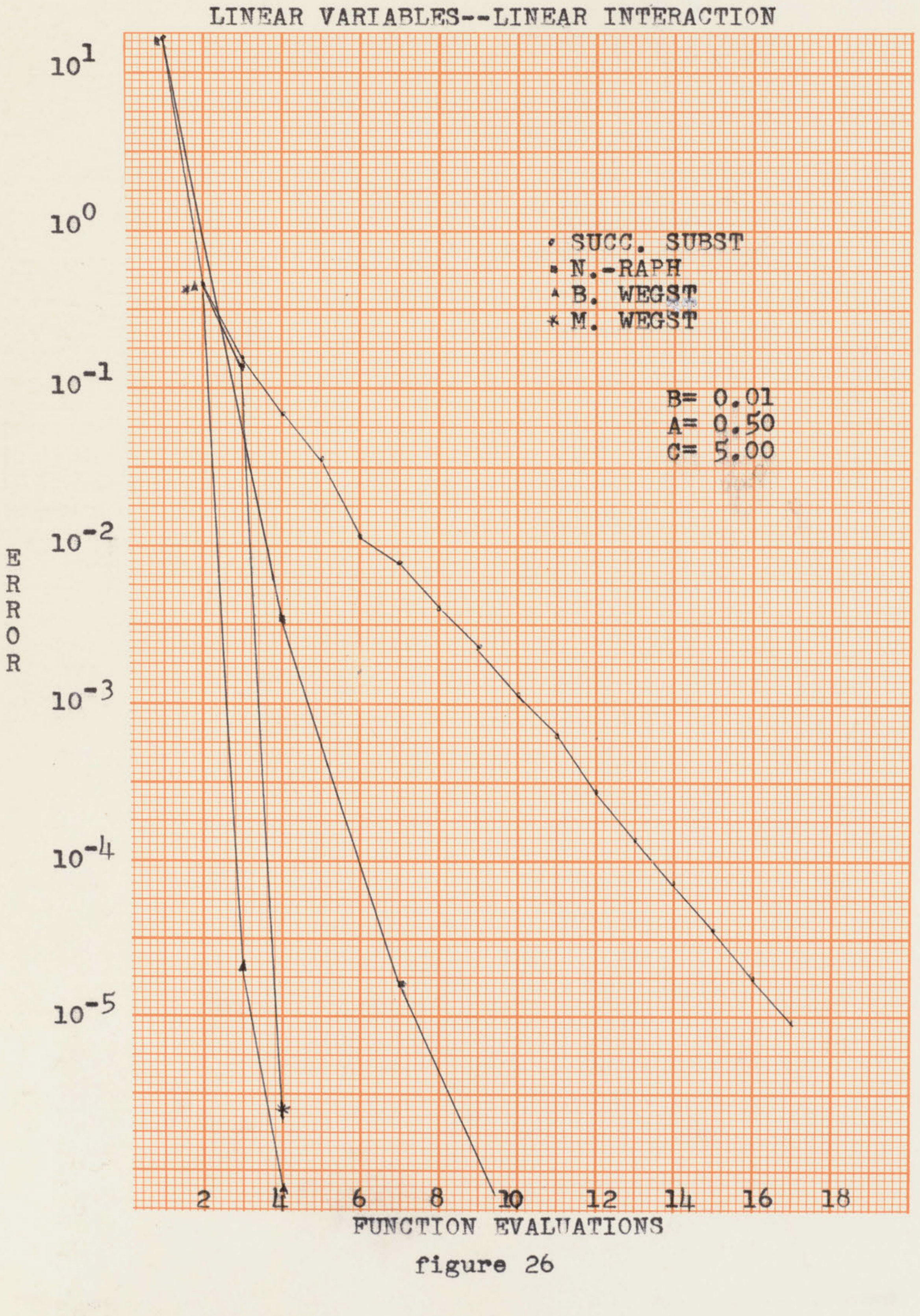
figure 24 "spot check"

LINEAR VARIABLES--LINEAR INTERACTION



FUNCTION EVALUATIONS

figure 25



LINEAR VARIABLES--LINEAR INTERACTION
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
ERROR
• SUCC. SUBST
▪ N.-RAPH
▴ B. WEGST
* M. WEGST
B= 0.01
A= 0.50
C= 5.00
2
4
6
8
10
12
14
16
18
FUNCTION EVALUATIONS

figure 26

lin / lin B= 1.0 10^{-2}

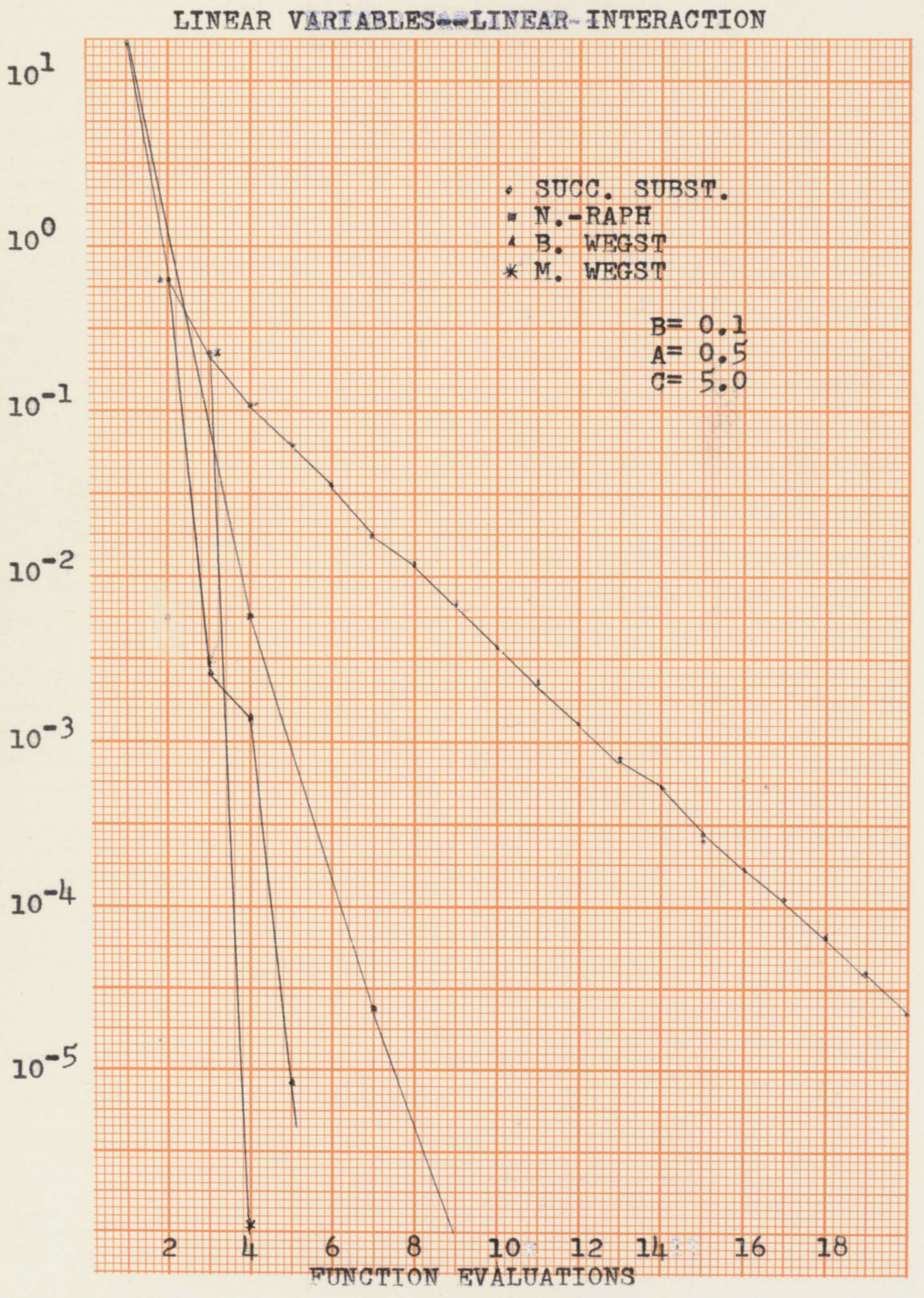

LINEAR VARIABLES--LINEAR INTERACTION
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
ERROR
2
4
6
8
10
12
14
16
18
FUNCTION EVALUATIONS
SUCC. SUBST.
N.-RAPH
B. WEGST
M. WEGST
B= 0.1
A= 0.5
C= 5.0

figure 27

lin / lin B = 0·1

LINEAR VARIABLES--LINEAR INTERACTION

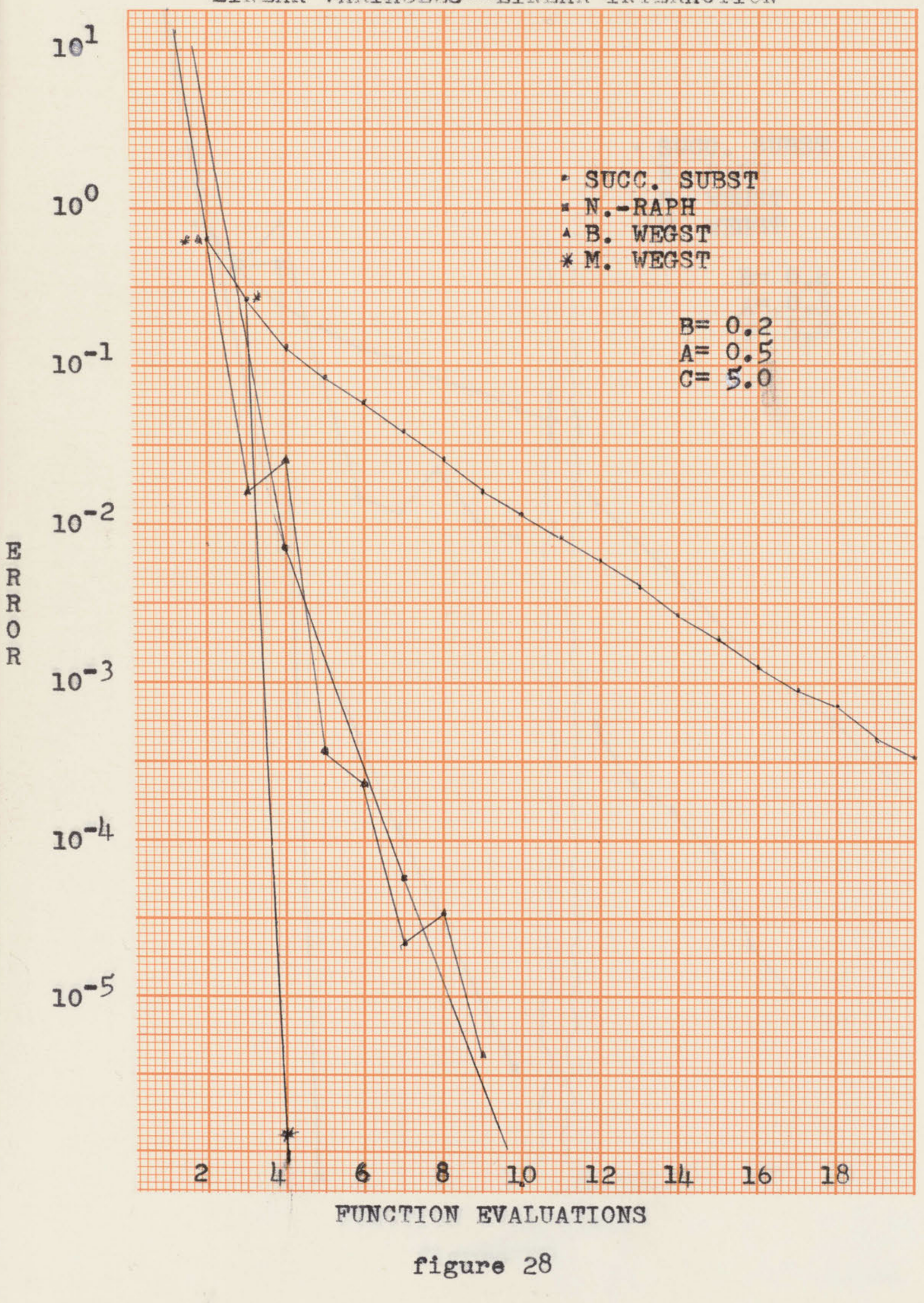


figure 28

lin /lin B= 0-2

LINEAR VARIABLES--LINEAR INTERACTION

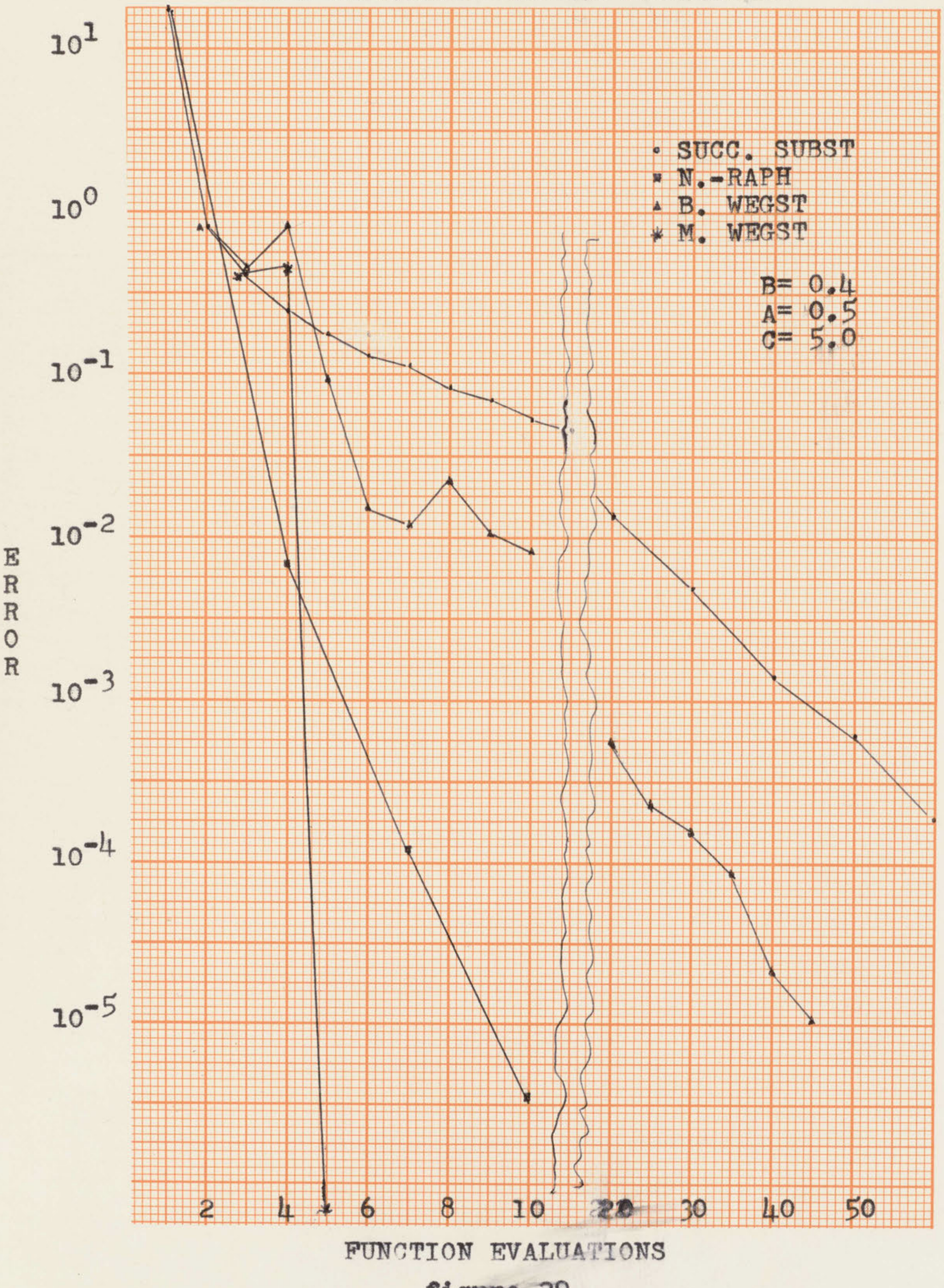


figure 29

lin/lin B = 0.110

LINEAR VARIABLES--LINEAR INTERACTION

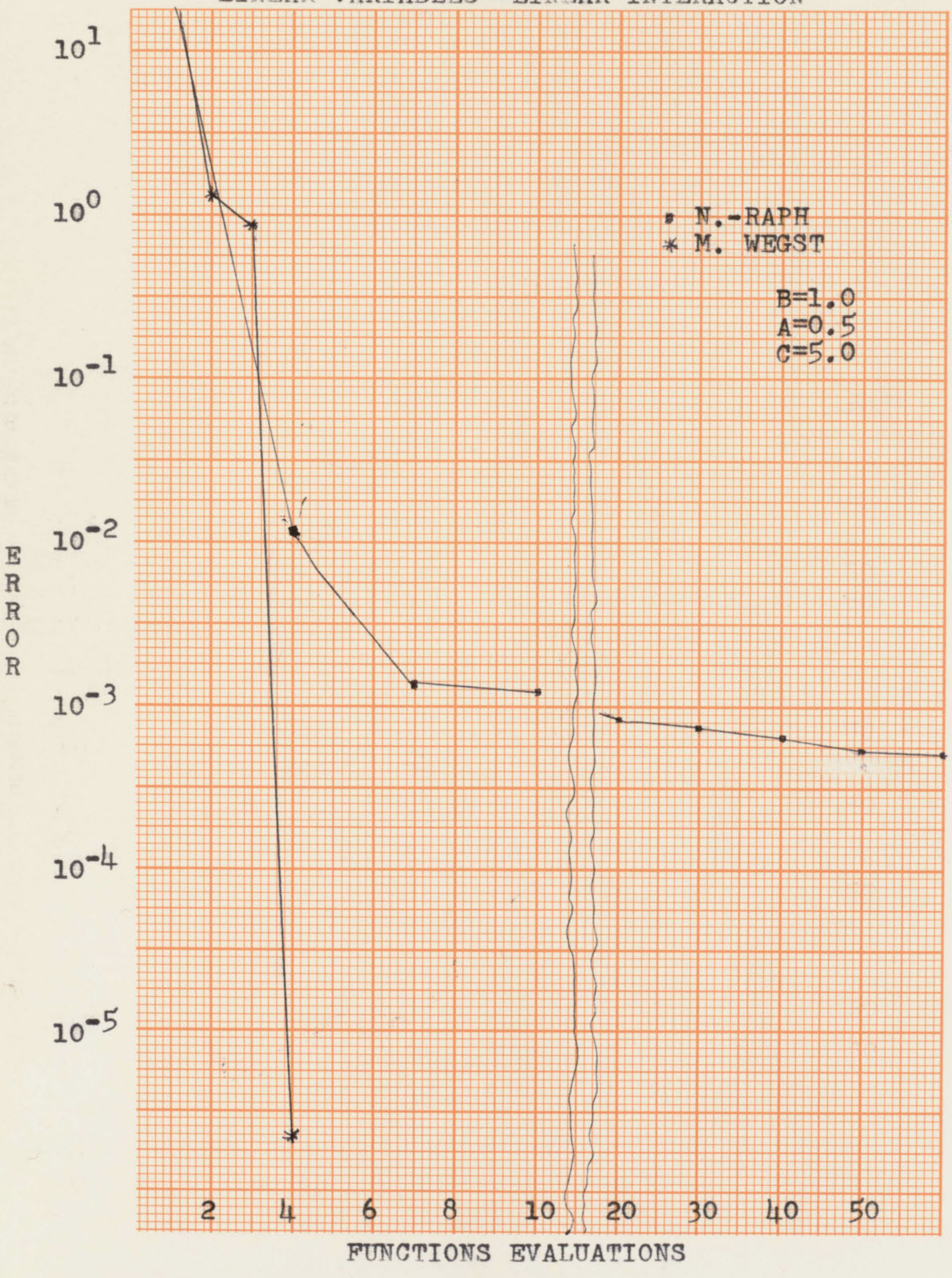
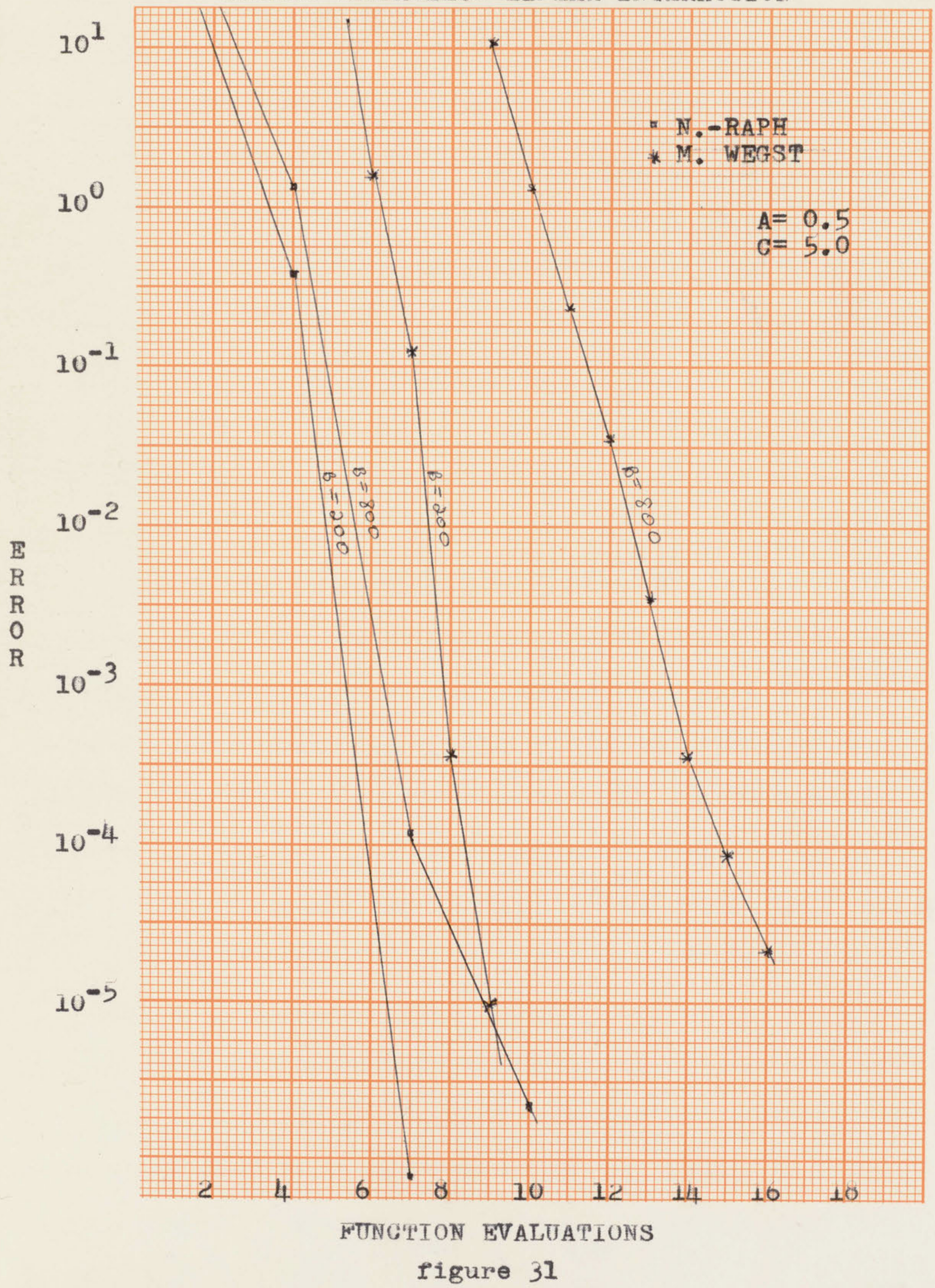


figure 30

lin/lin B = 1.0

LINEAR VARIABLES--LINEAR INTERACTION



FUNCTION EVALUATIONS

figure 31

interaction is increased, its rate of convergence decreases. Finally it becomes very erratic and eventually fails to solve the problems for values of B greater than 0.4.

The Newton-Raphson Method appears to be very good in solving a large class of these two-dimensional problems. It was able to solve problems for values of B up to 1,000,000. This method did, however, run into trouble for the value of B=1.0. Here, the procedure assymptoted quickly. The main problem with this method is that it needs N function evaluations for each iterative step, thus slowing it down drastically. One probably would not to use this method for small to moderate values where other methods are fairly good.

The Multidimensional Wegstein Method works very well for values of B up to about 3000, where its slope levels out; AND, finally, the procedure diverges. The degree of interaction needed to cause this method to diverge, might, however, be lowered by its use of three successive Substitution iterations before the procedure takes over. These steps send the procedure far from the solution; and, for values of B great enough, it is unable to recover, and fails to return. If this is the case The Multidimensional Wegstein Method may be able to solve "tougher" problems by using a different algorithm to generate the first N iterations(e.g. perturbations of the initial values). As it is presently programmed, however, it works very well for a wide range of interactions.

In order to test the sensitivity of the methods to intravariable interactions, I set $B= 0.1$, and $C= 5.0$, then ran a "spot" check for each of the methods to see how high of a value of A is needed before the method can no longer solve the problem. For this check, I used a convergence criterion of 0.01 both for the relative and the absolute error tolerances, and starting values of 0.25 and 0.75 for the two variables. See figure 32 for the results of the "spot" check. Based on these results, I chose values of $A= 1.25_{10}-6$, 0.5, 0.7, 0.99, 5.0, and 100 for closer examination. Graphs of error versus number of function evaluations (figs. 33-37) were obtained for these values of A.

Successive Substitution works well for small amounts of intra-variable interaction, $A \leq 0.5$. As the value of A gets moderately large, though, the method rapidly levels out. For $A= 0.99$, it approaches an assymptote at an error of about .01. For larger values of A it diverges.

The Bounded Wegstein Method seems very sensitive to the size of A. For A small to moderate, the method works very well. As the value of A becomes large enough to make the eigenvalues of the system greater than 1.0 at the solution, the method becomes much slower. For values that make   1.0, the method diverges.

The Newton-Raphson method appears very stable to the value of A. It obtains solutions in the same amount of function evaluations for values of A from 0 to 100. The method is slowed somewhat by the (N+1) function evaluations needed for each iteration. This affect would become more drastic for systems of equations involving more variables.

The Multidimensional Wegstein Method was, again, consistently better than any of the other three methods. For small values of A, this was not very evident, for all the methods were working very well. But, as the size of A increases, both the Newton-Raphson and the Multidimensional Wegstein Method become substantially better than the other two methods. For very large values of A, the Newton-Raphson and the Multidimensional Wegstein method get better. Since the Multidimensional Wegstein method requires a lot more computational time than the Newton-Raphson Method, the latter might be preferred in this range. I suspect, however, that the Newton-Raphson Method will not work so well for equations involving more variables.

A	SUCC..SUB	B. WEG	N.-RAPH	M. WEG
$1.25_{10}{-6}$	2	3	4	5
$2.5_{10}{-6}$	2	3	4	5
0.5	8	3	4	5
0.7	14	3	4	5
0.9	99	41	2	¢
0.99	*	*	4	6
2.5	#	#	4	5
5.0	#	#	7	5
100	#	#	10	5

¢-matrix inversion incomplete
*-failed to converge in 100 iterations
#-method diverged

figure 32

LINEAR VARIABLES--LINEAR INTERACTION

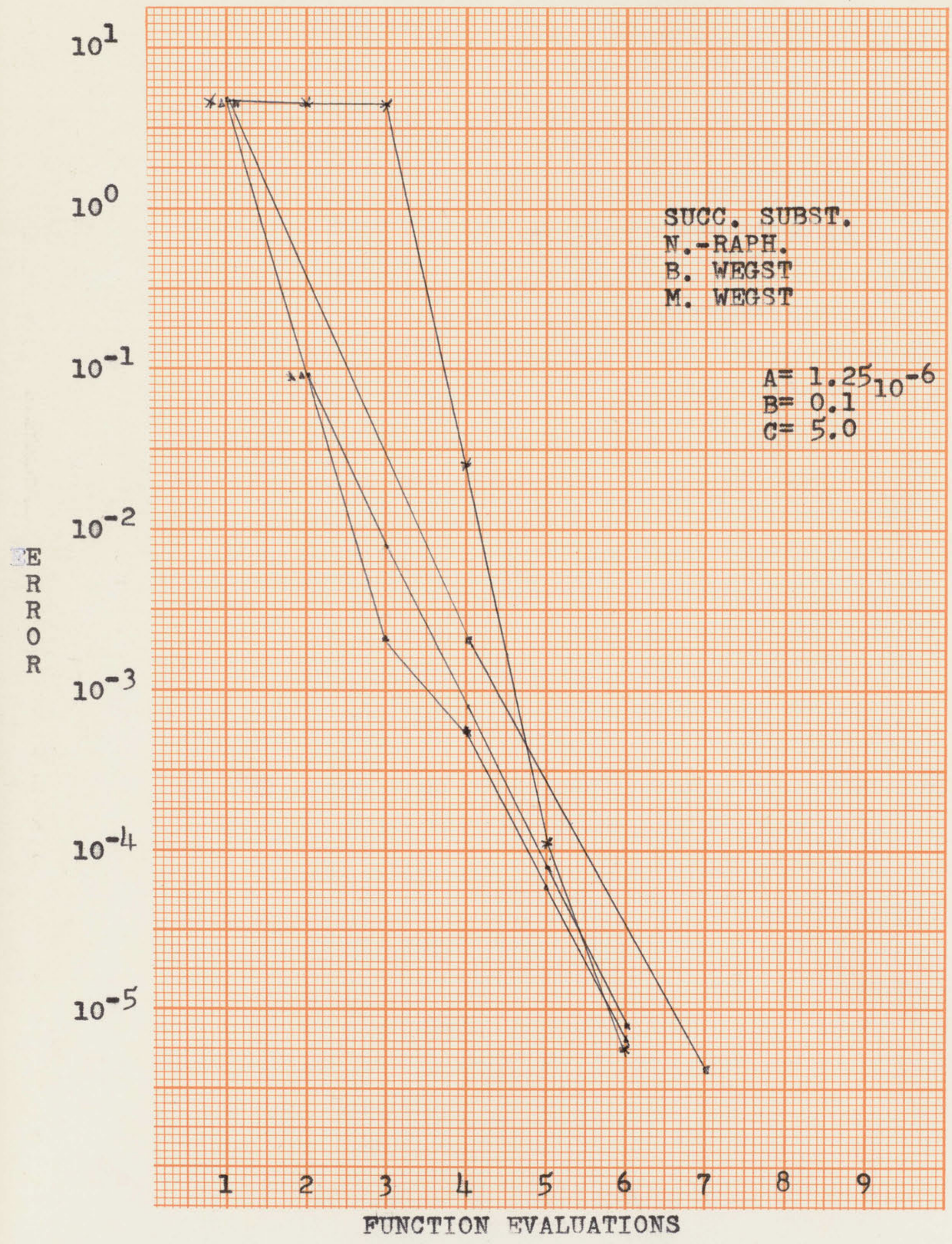


figure 33

LINEAR VARIABLES--LINEAR INTERACTION

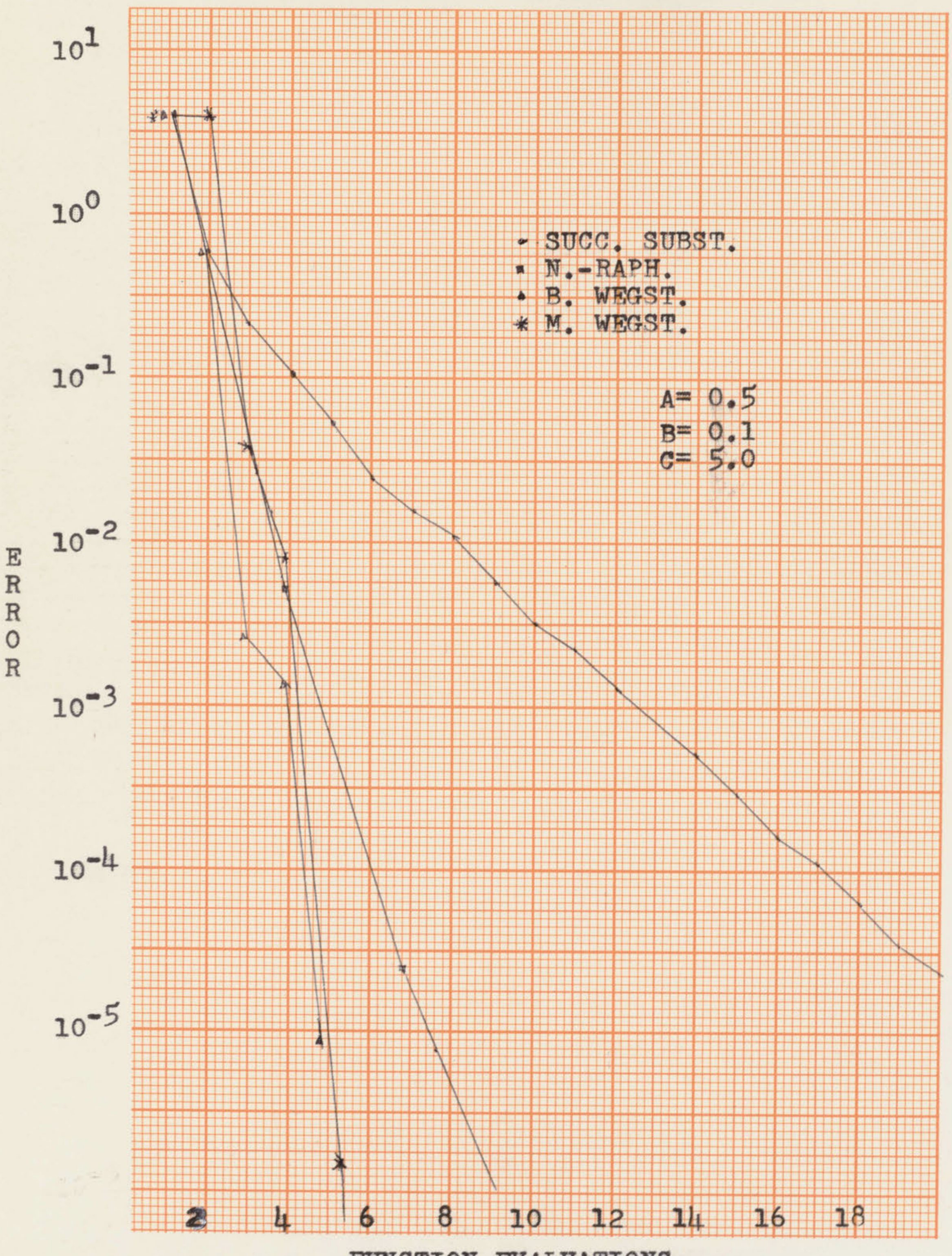


figure 34

LINEAR VARIABLES--LINEAR INTERACTION

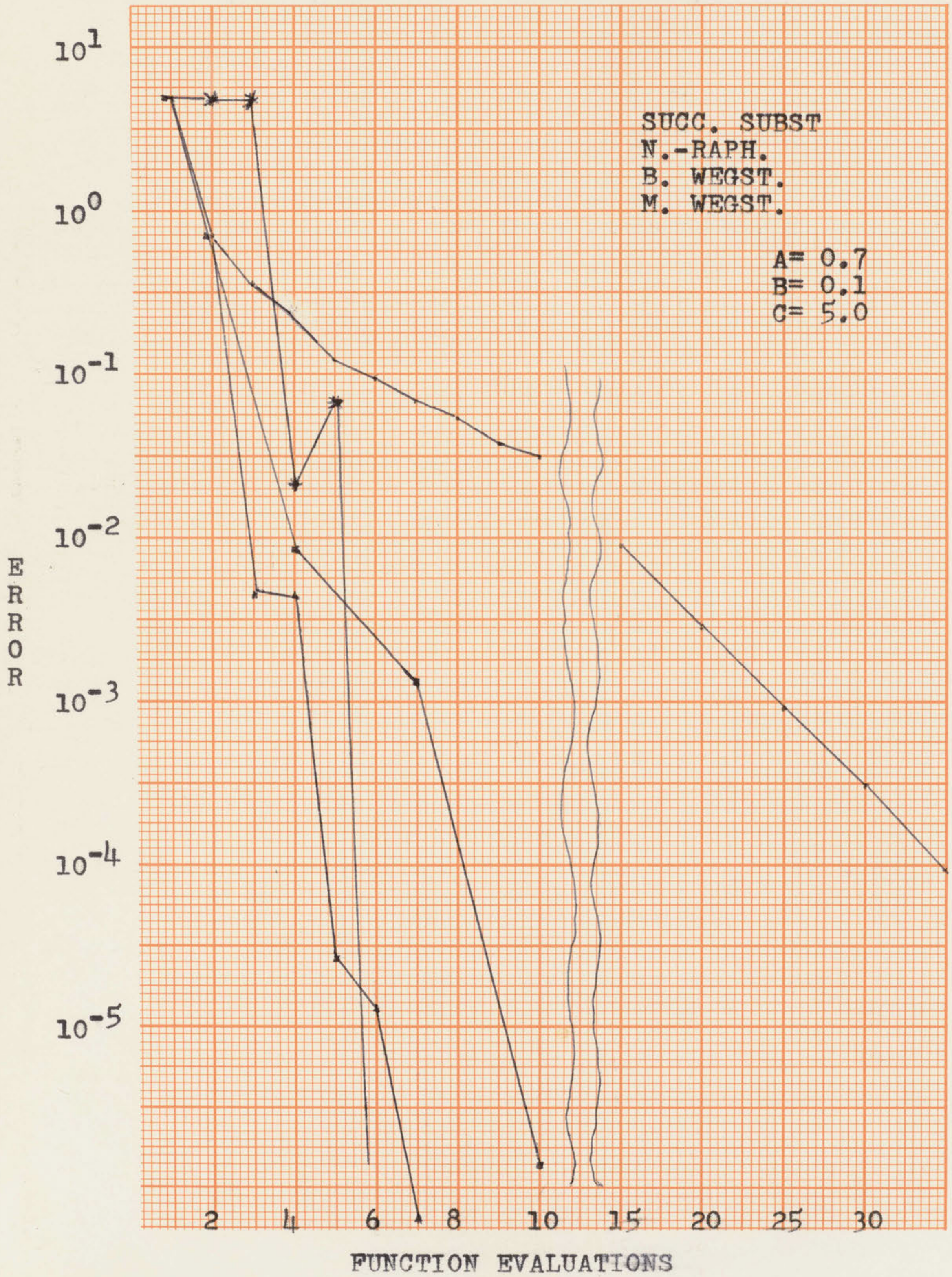


figure 35

LINEAR VARIABLES--LINEAR INTERACTION

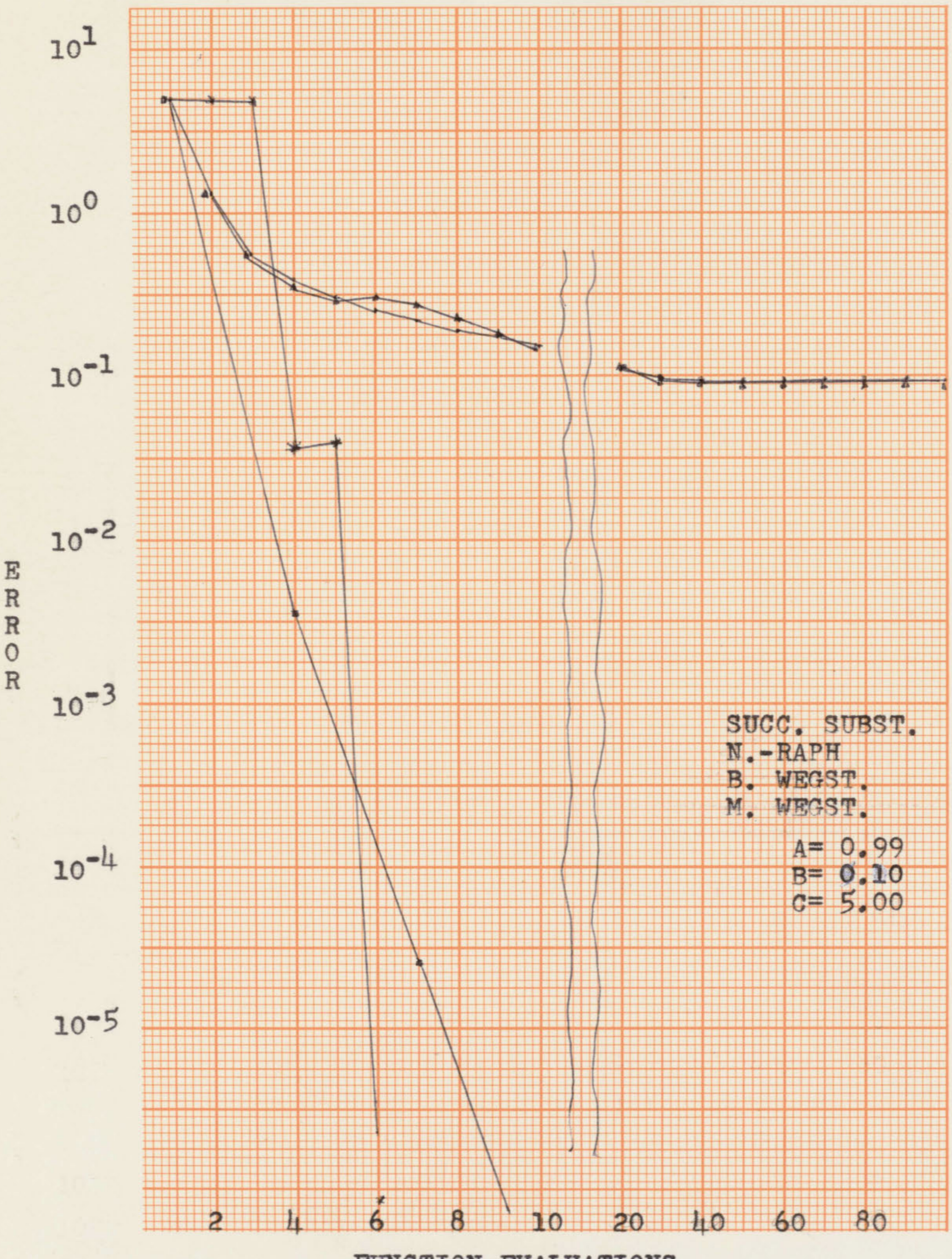


figure 36

LINEAR VARIABLES--LINEAR INTERACTION

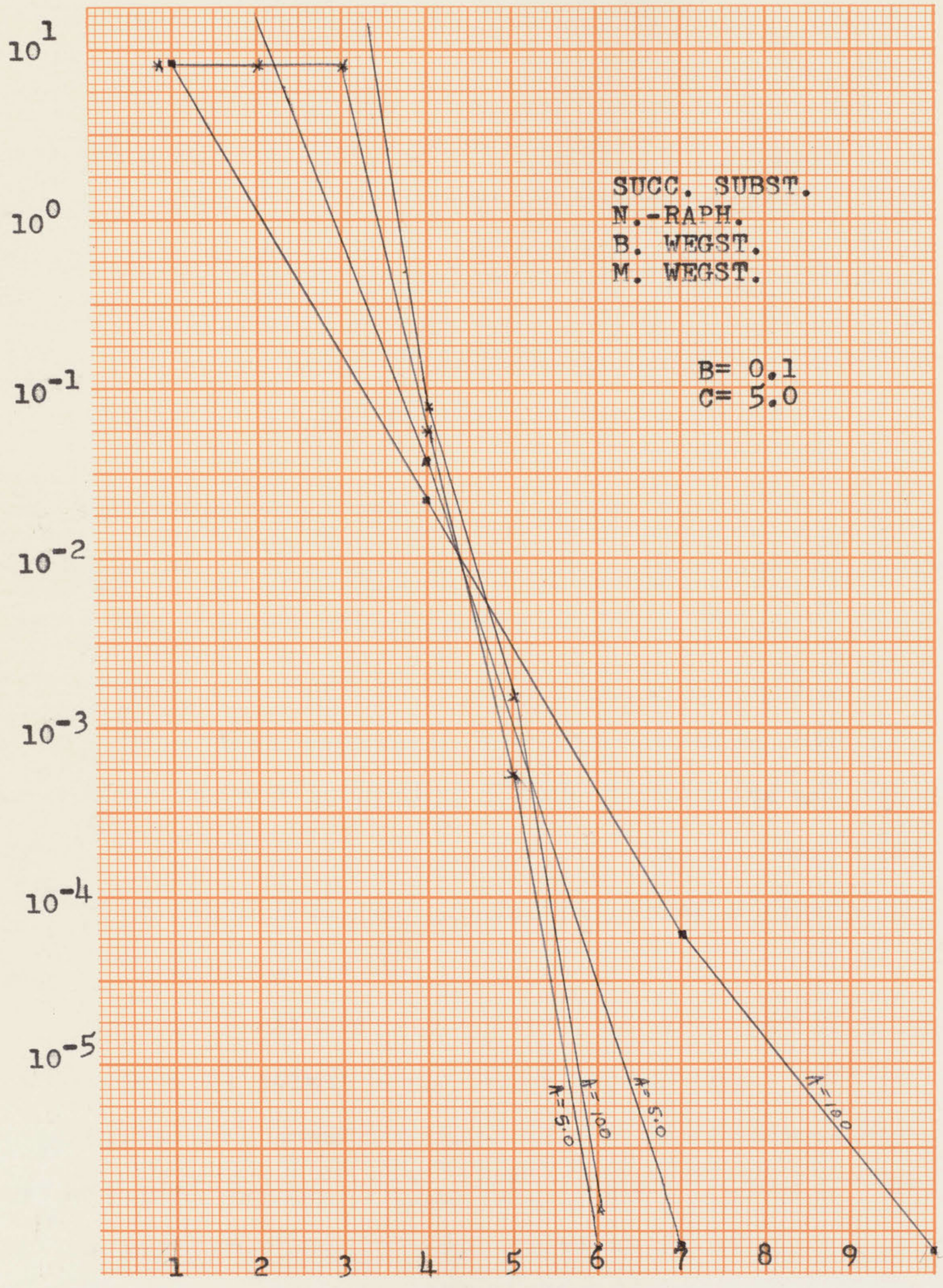


figure 37

B: Linear Variables, Non-Linear Interaction:

To determine if non-linearity of the interaction would have any affect as to the number of function evaluations needed to solve the problems and as to the range of problems solved by the various procedures, I chose this set of equations:

$$x_1 = Ax_1 + Bx_2^2 + C$$
$$x_2 = Ax_2 + Bx_1^2 + C$$

The solution to this set of equations is $x=((1-A)\pm \mathrm{sqrt}(((A-1)^2 - 4BC))/2B$. I again set $A= 0.5$ and $C=5.0$, and varied the value of B, the amount of interaction. The solution now becomes: $x= (0.5\pm \mathrm{sqrt}(0.25-20B))/2B$, and is not real for values of $B\geq 0.0125$. In order to determine where each procedure ran into problems, I set the absolute and relative errors tolerances to 0.01, and solved the problems using the various values of B. The initial values of x were: $x_1=0.25$, $x_2= 0.75$ (see figure 8).

All of the methods were very efficient right up to the point where the solution no-longer exists, so I decided to investigate this area further, and to extend the search into the negitive region(see fig. 39). Since the Newton-Raphson and the Multidimensional Wegstein Methods ran into problems between B=-1.0 and B= -10,000, I also expanded in this range (see fig. 40). Based upon these "spot" checks, I chose these

B	0.0	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{0}
S.SUBST.	7	7	7	7	11	*
B.WEGST.	3	3	3	3	6	*
N.-RAPH.	4	4	4	4	7	*
M.WEGST.	4	4	3	3	5	*

*-procedure failed to converge

figure 38

B	.011	.012	.0125	-10^{-6}	-10^{-4}	-10^{-2}	-1.0	-10^{2}	-10^{4}
S.SUB.	12	14	16	6	6	5	#	#	#
B.WEG.	7	7	8	3	3	4	#	#	#
M.WEG.	5	8	7	4	3	5	10	*	#
N.-RAP.	10	10	10	13	13	10	31	13	*

#-system diverged
*-matrix inversion incomplete

figure 39

B	-25	-50	-75	-200	-500
N.-RAPH.	10	13	13	7	*
M.WEGST.	4	22	*	*	*

*-matrix inversion incomplete

figure 40

values of B for further analysis: 0.0, 0.01, 0.011, 0.0125, -0.01, -1.0, -50). The results are plotted on figs. 41- 46.

The Method of Successive Substitution consistently required more function evaluations than any of the other methods. The Range of problems that it could solve was also very limited. It would only solve problems whose values of B ranged between 0.0125 and -0.1, the Newton-Raphson and the Multidimensional Wegstein Methods appear to solve equations for which B can be up to about -100.

The Bounded Wegstein Method has the same range problem as the Method of Successive Substitution, though it solves problems with very little interaction in much fewer function evaluations than does the Method of Successive Substitution.

The Newton-Raphson and the Multidimensional Wegstein Methods both solve a wider range of problems than the other two methods, and solve the problems in far fewer iterations. The Newton-Raphson method requires less time per function evaluation than does the Multidimensional Wegstein Method, so that if function evaluations do not require very much time, then the Newton-Raphson method would be preferred. If, however, function evaluations require much more time than the iteration itself, the Multidimensional Wegstein Method would be preferred.

LINEAR VARIABLES--NON-LINEAR INTERACTION

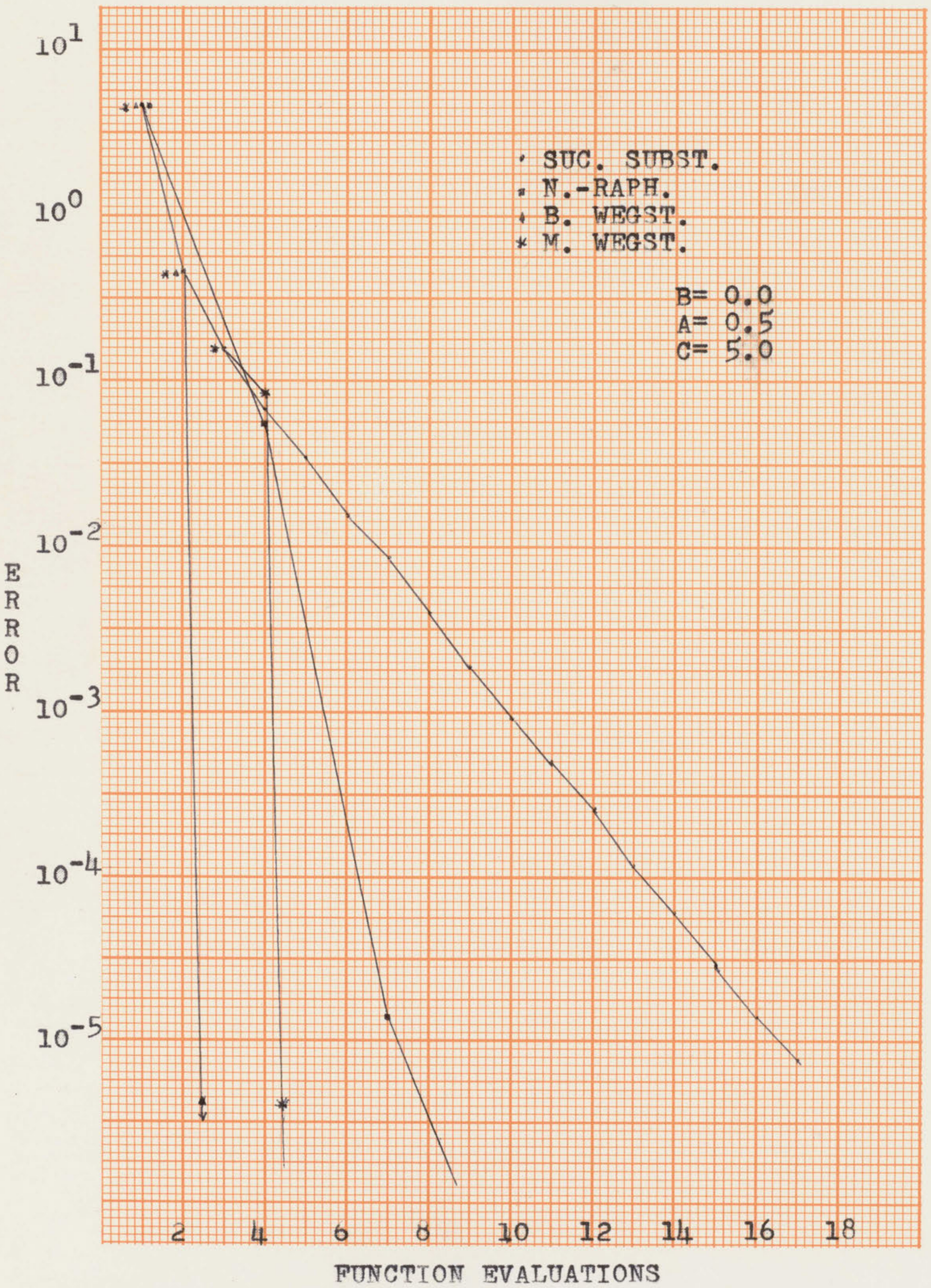


figure 41

LINEAR VARIABLES--LINEAR INTERACTION

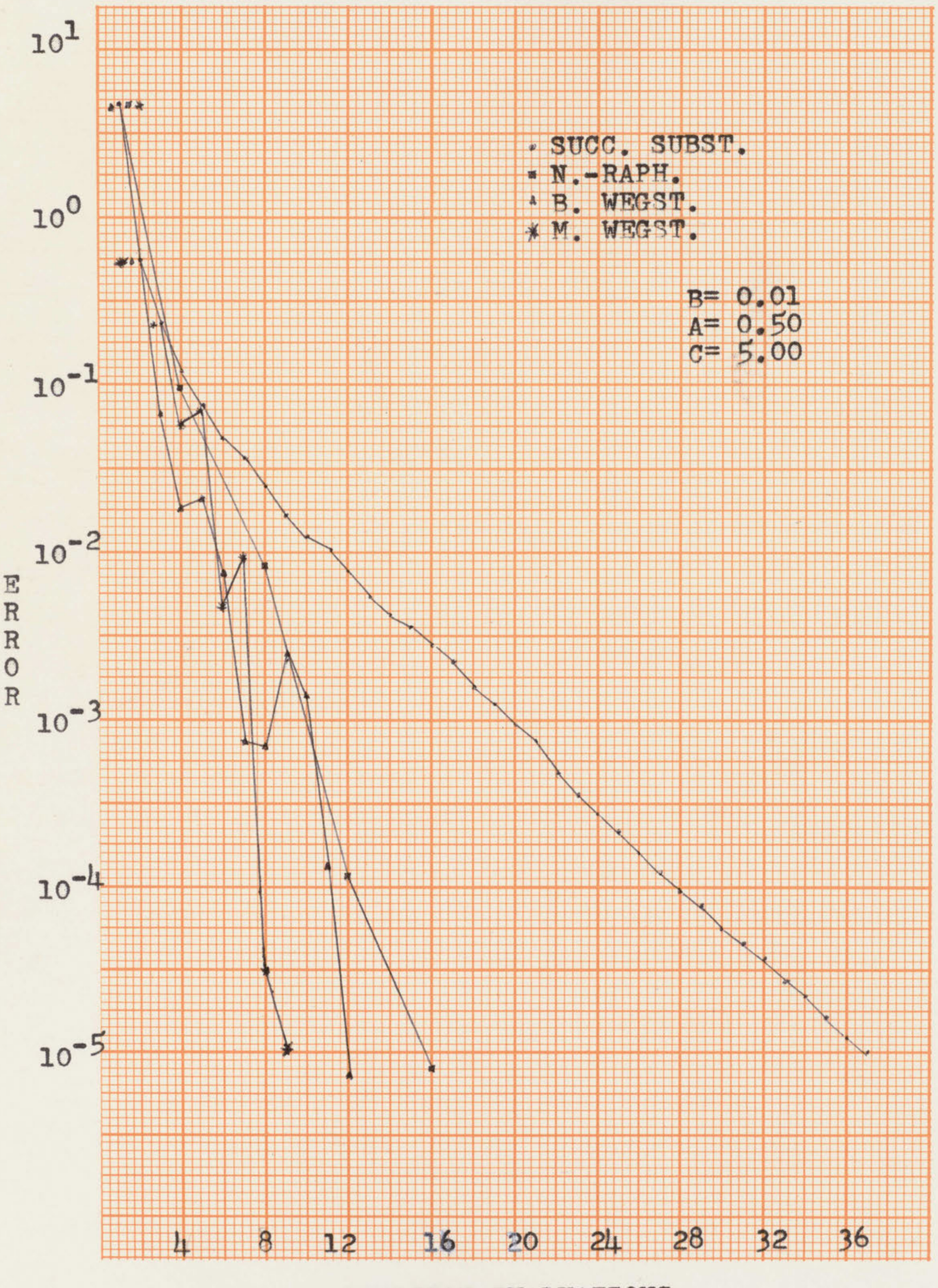
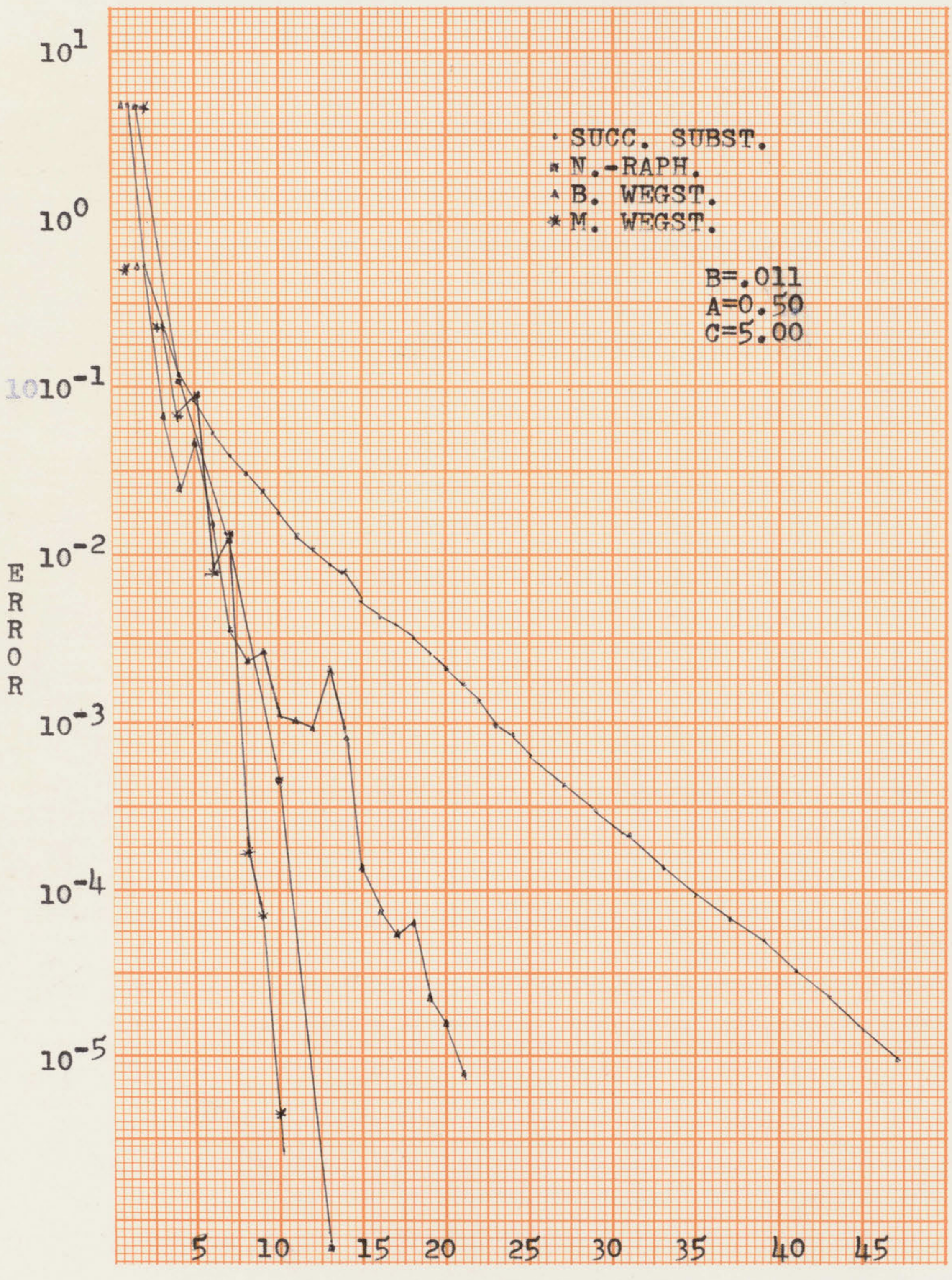


figure 42

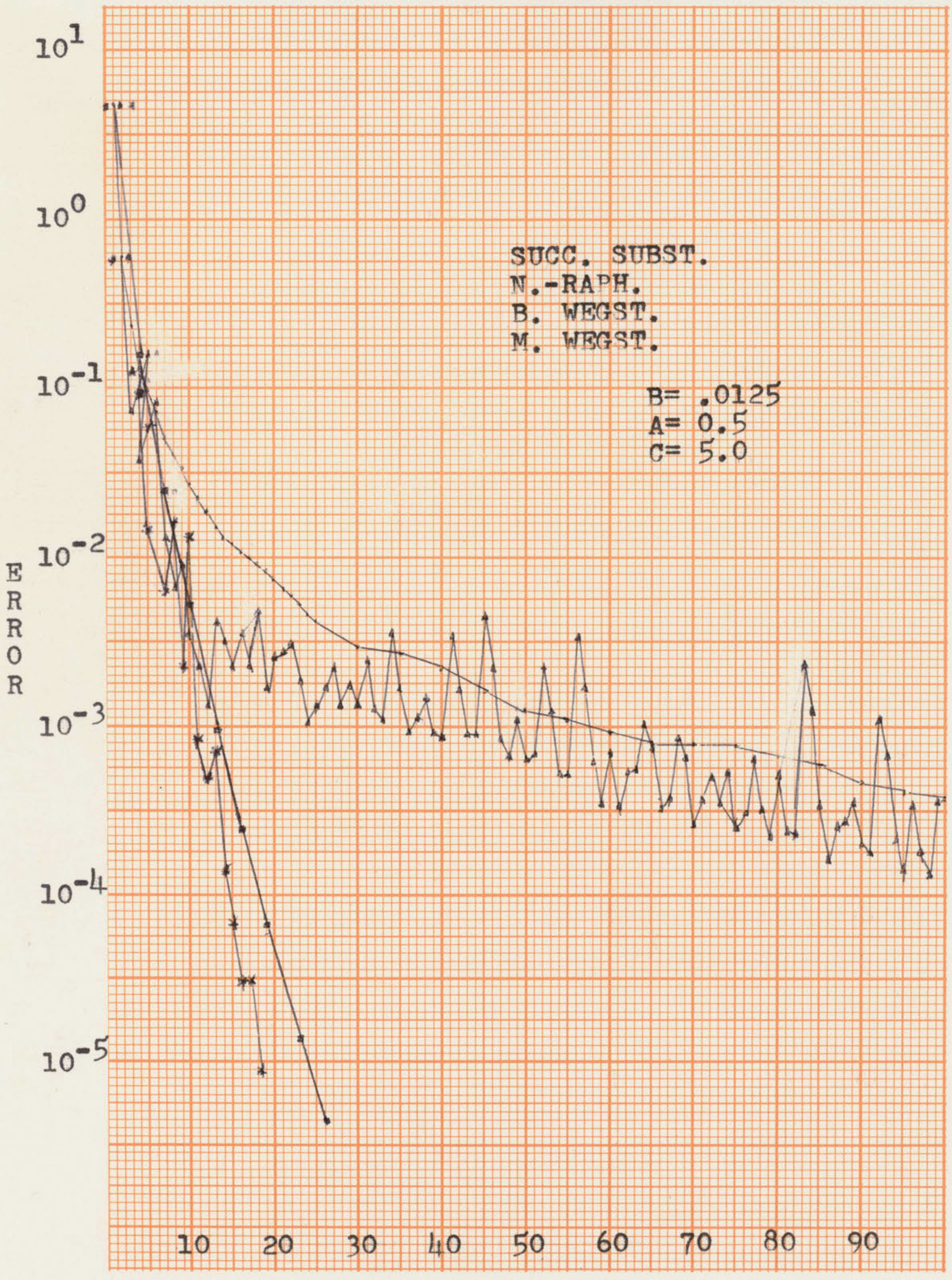
LINEAR VARIABLES--NON-LINEAR INTERACTION



FUNCTION EVALUATIONS

figure 43

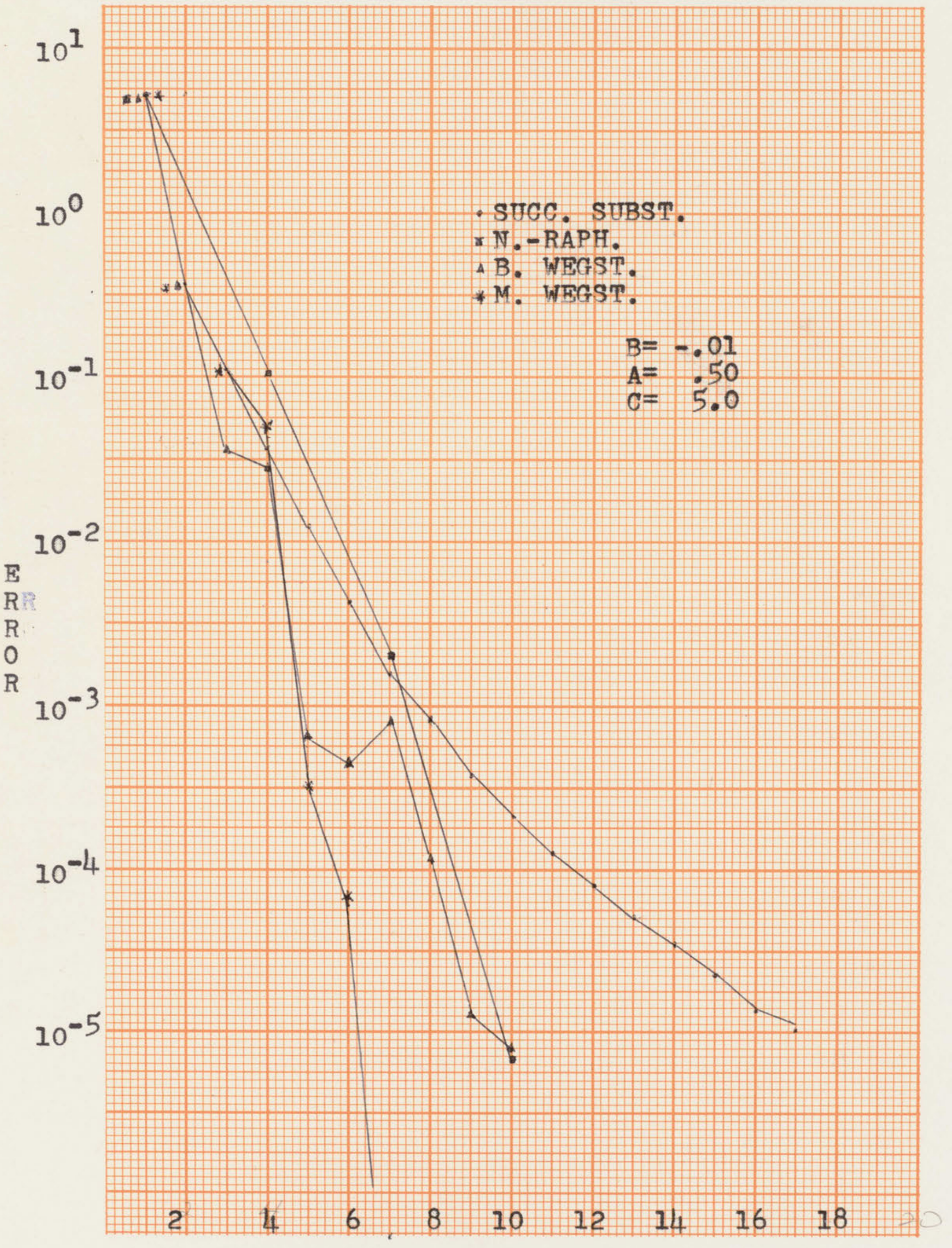
LINEAR VARIABLES--NON-LINEAR INTERACTION



FUNCTION EVALUATIONS

figure 44

LINEAR VARIABLES--NON-LINEAR INTERACTION

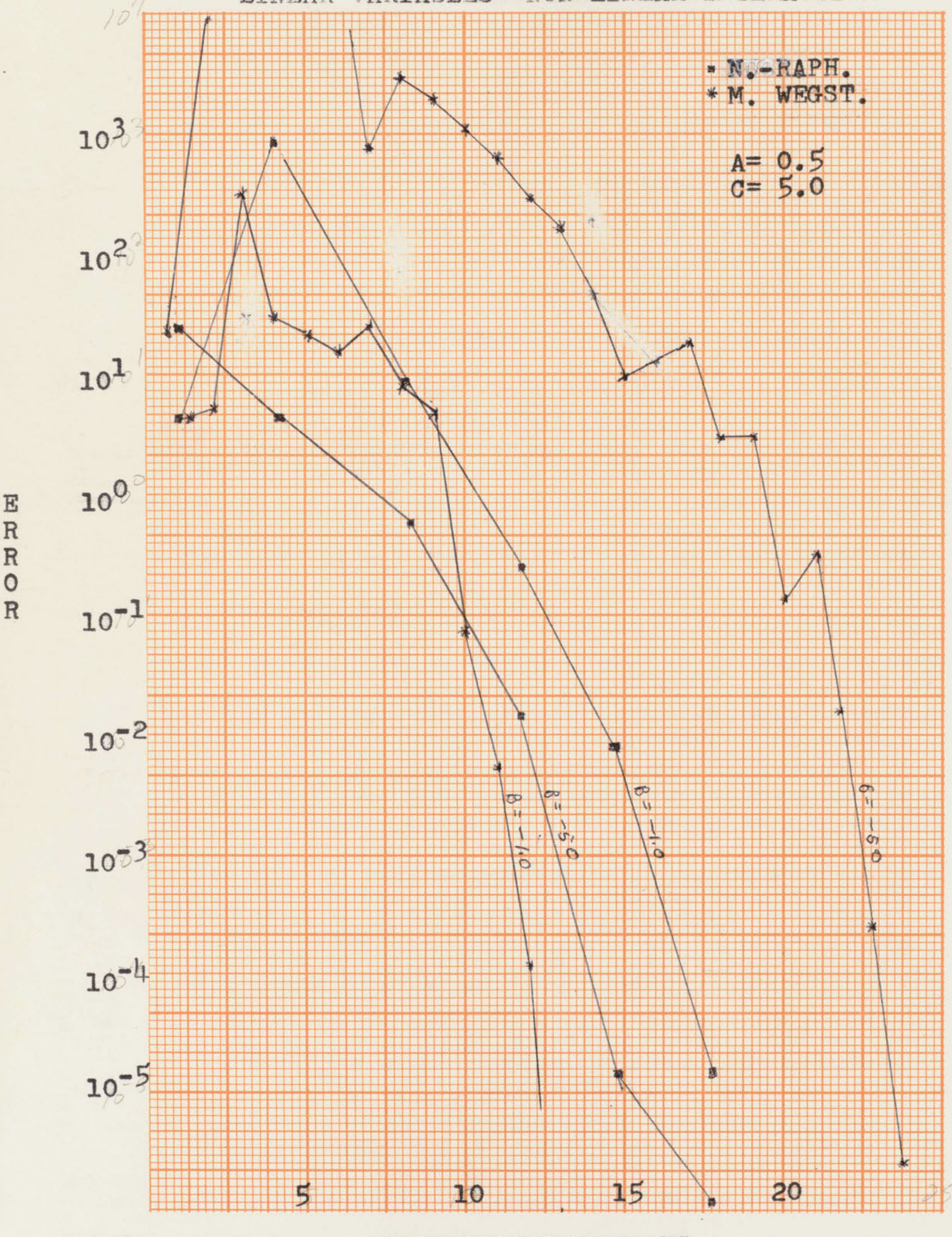


FUNCTION EVALUATIONS

figure 45

B = -.01

LINEAR VARIABLES--NON-LINEAR INTERACTION



FUNCTION EVALUATIONS

figure 46

Again, to determine the affect of intra-variable interactions on the rate of convergence for the case of linear variables with non-linear interaction, I now set B= 0.1, and C=5.0. From the equation for the solution to this problem, one can see that the solution is real for all negative values of $A \leq -.414$, and for positive values of A greater than 2.414. The eigenvalues of the solution are only less than one for a very narrow interval, $-.717 \leq A \leq -.414$, and for $2.414 \leq A \leq 2.717$. I now did a "spot" check of values of A for which the methods will solve this problem(see fig. 47). Based on these tests, I chose the following values of A for further analysis: A= -0.5, 5.0, 100(see figs. 48-49).

Even though the eigenvalues were less than one at the solutions for three of these test runs, the methods of Successive Substitution and the Bounded Wegstein method failed to obtain solutions to these test runs. This probably resulted because the eigenvalues at the test solution were greater than one.

The Newton-Raphson and the Multidimensional Wegstein Method again worked very well for small values of A. When the value of A gets very large, however, the Newton Raphson Method becomes even better than the Multidimensional Method. The Multidimensional Wegstein Method fails for values of A greater than 10,000, while the Newton-Raphson Method still works well.

A	S. SUBST.	N-RAPH.	B. WEG.	M. WEG.
-.7	#	10	#	5
-.5	#	10	#	4
2.5	#	10	#	8
5.0	#	7	#	7
10^2	#	7	#	10
10^4	#	7	#	#

#-system diverged

figure 47

LINEAR VARIABLE--NON-LINEAR INT.

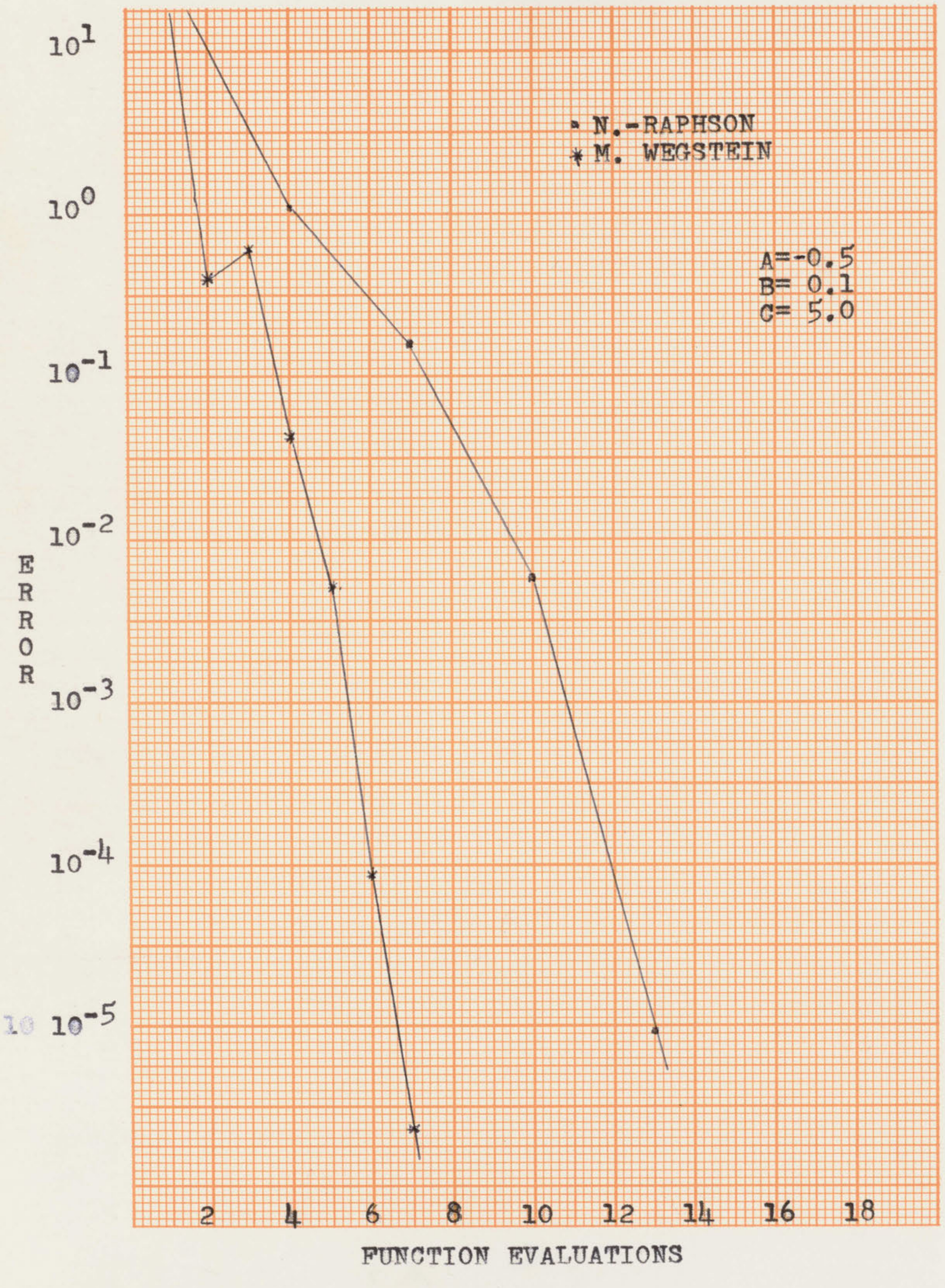


figure 48

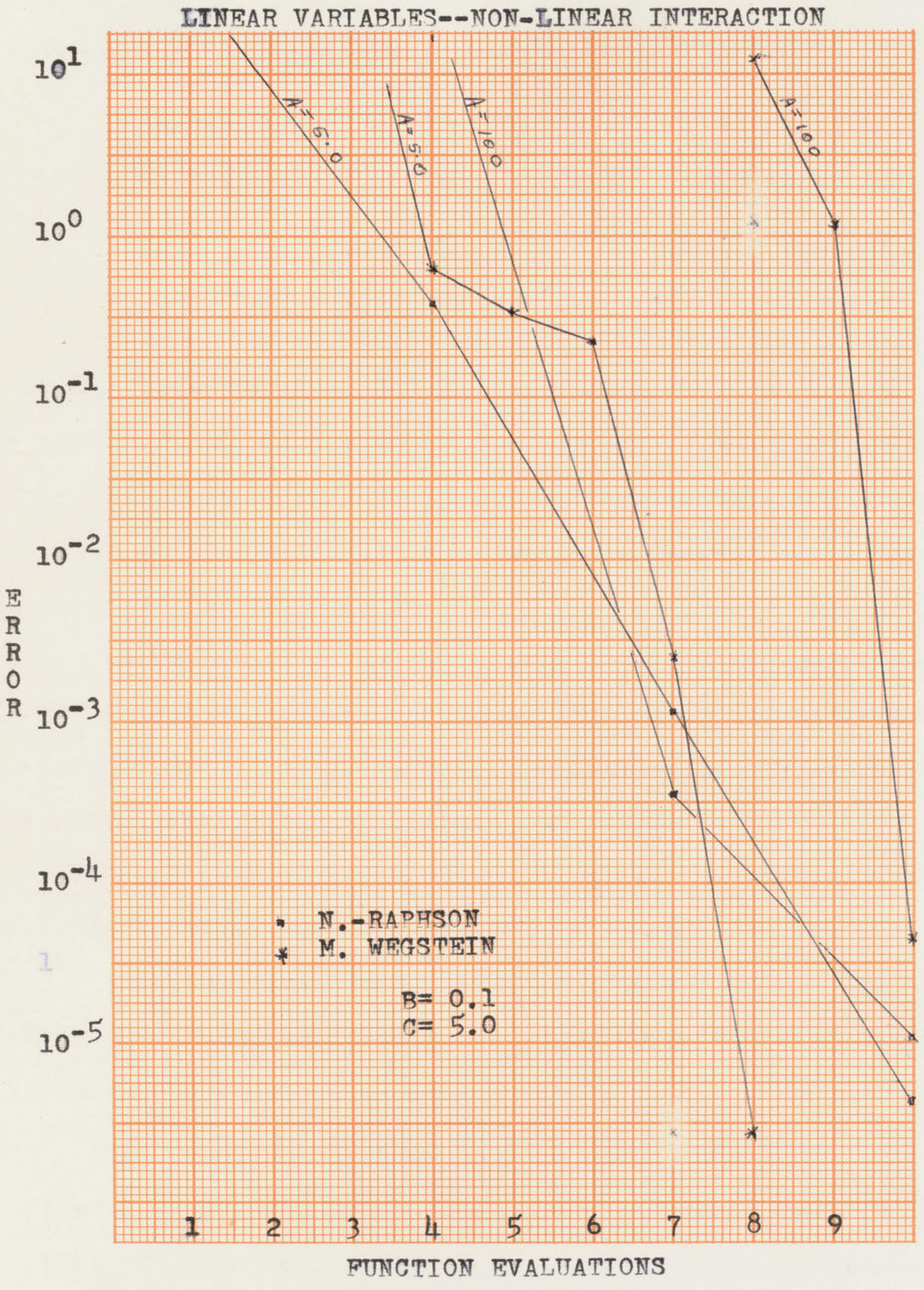

LINEAR VARIABLES--NON-LINEAR INTERACTION
10^1
10^0
10^-1
10^-2
10^-3
10^-4
10^-5
ERROR
A=6.0
A=5.0
A=100
A=100
N.-RAPHSON
M. WEGSTEIN
B= 0.1
C= 5.0
1
2
3
4
5
6
7
8
9
FUNCTION EVALUATIONS

figure 49

C: NON-LINEAR VARIABLES, LINEAR INTERACTION.

To investigate the case of non-linear variables, with linear interaction, I chose this set of equations:

$$x_1=Ax_1^2+ Bx_2+ C$$
$$x_2=Ax_2^2+ Bx_1+ C$$

Again, by setting A= 0.5, and C= 5.0, one can determine the sensitivity of solution attainment to the amount of interaction, B. First I did a broad "spot" check by varying the value of B between 5 and 10^4, and setting the convergence tolerance to 0.01, see fig. 50.

The Method of Successive Substitution and the Bounded Wegstein Method both fail to obtain proper solutions for any of the values of B, while the Multidimensional Wegstein Method ran into trouble for values of B between 100 and 1000. I, therefore, picked several values of B between these numbers and did another "spot check" of this range, see fig. 51.

Based on these "spot checks", I chose several values of of B for further analysis, B= 5, 50, 100, 150, 200. These runs are plotted on fig. 52.

The solution to this problem is $x=(1-B\pm\text{sqrt}((1-B)^2-4AC))/2A$. For A= 0.5, C= 5.0, this becomes $x= (1-B\pm\text{sqrt}((1-B)^2-10)$. So the solution is real for B greater than 4.162. The absolute value of the eigenvalues of these equations at the solution

B	S. SUBST	B. WEG	N.-RAP.	M. WEG
5	#	#	10	10
10	#	#	7	11
50	#	#	7	9
100	#	#	7	13
10^3	#	#	#	*
10^4	#	#	#	*
10^5	#	#	#	*

#- failed to converge to the solution
*- matrix inversion incomplete

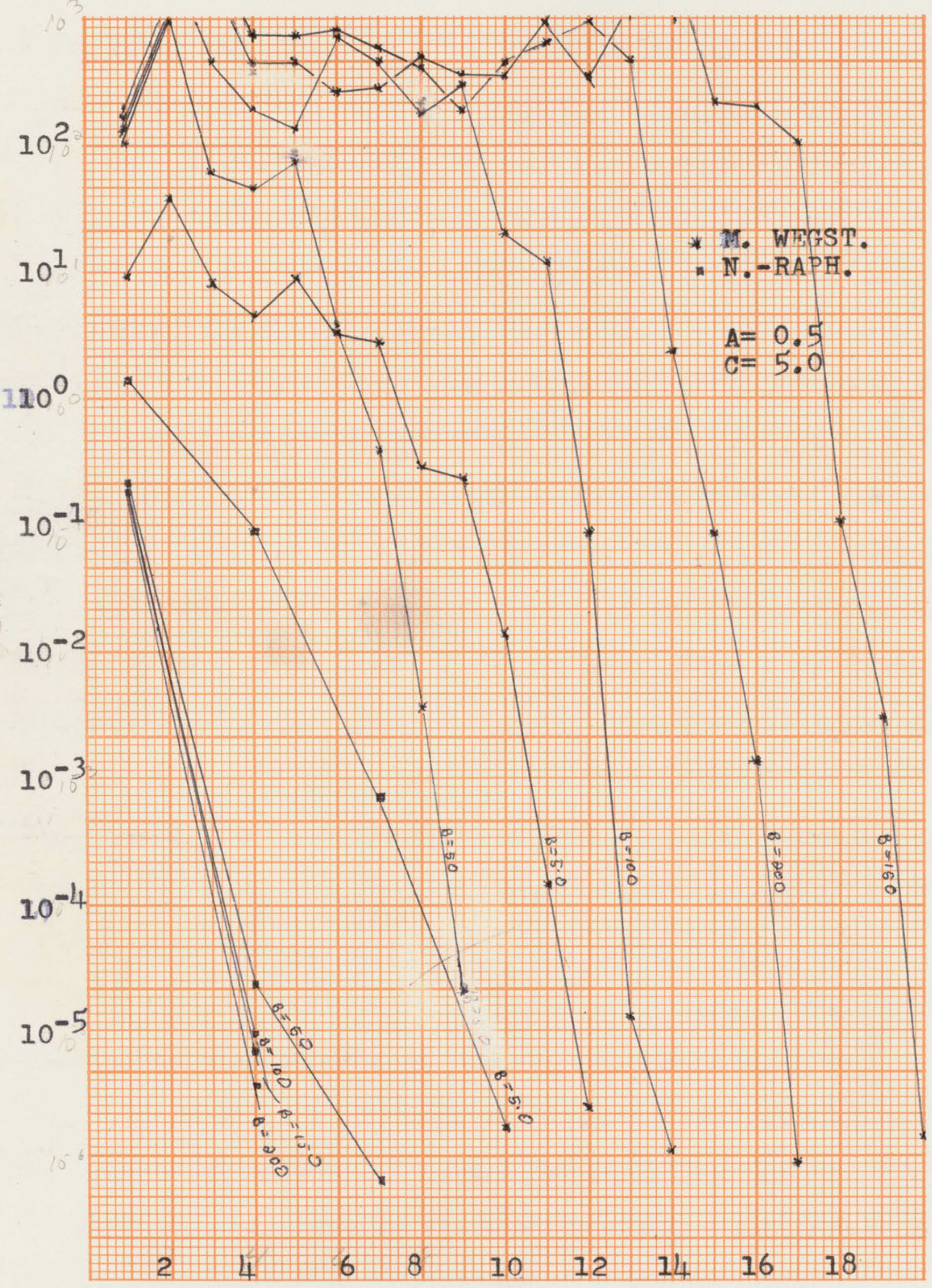
figure 50

B	N.-Raph.	M. WEG.
150	7	19
200	7	16
500	10	*
800	10	*
900	*	#

*-matrix inversion incomplete
#-system diverged

figure 51

NON-LINEAR VARIABLES--LINEAR INTERACTION



FUNCTION EVALUATIONS

figure 52.

are greater than one for all values of B, therefore, the Method of Successive Substitution was unable to solve any of the problems. The Bounded Wegstein Method also failed consistently. The Multidimensional Wegstein and the Newton-Raphson Methods were able to solve these problems successfully for values of B up to 200.

Again the Newton-Raphson method was slightly better than the Multidimensional Wegstein Method. Both methods get slower as the lower bound of B is approached (4.162). The Multidimensional Wegstein Method gets much better as it approaches the solution, and appears to be as good as the Newton-Raphson method once it gets near the solution. Apparently the first two successive substitution iterations send the method far away from the solution, and it takes a long time for the method to recover. In this case the Multidimensional Wegstein Method would be much better if the first N iterations were made by perturbing each variable slightly.

In order to test the sensitivity of each convergence procedure to the amount of intra-variable interaction, I again set B= 0.1, and C= 5.0. The solution is now defined for values of A less than .045. I, therefore, ran a series of tests, using values of A varying between 0.0 and 0.045 to see how well each of the methods performed over this range. See figs. 53-55.

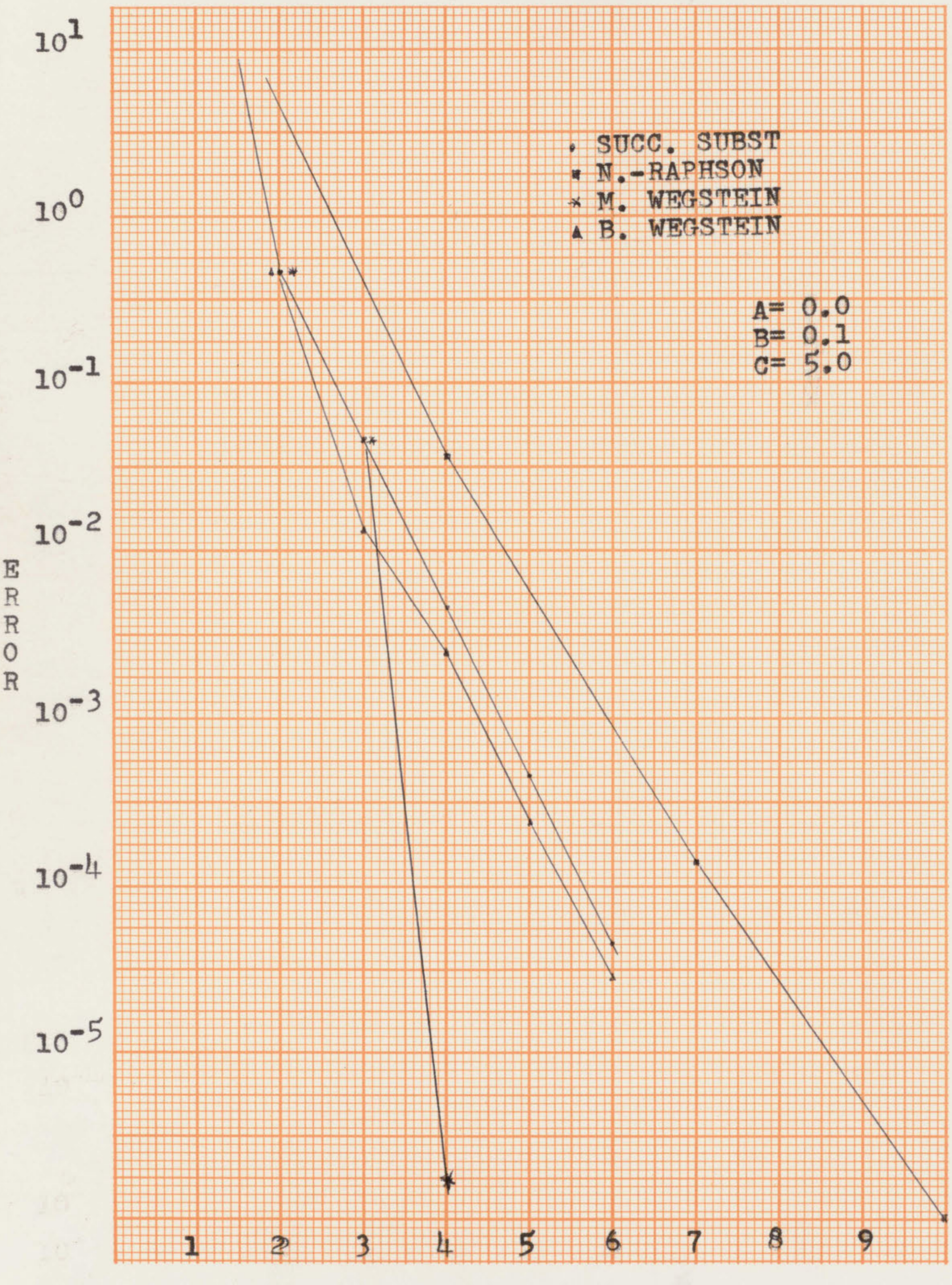
The Multidimensional Wegstein Method again performed better than any of the other methods over this range.

The Newton-Raphson Method performed poorly for A=0.0, but as the value of A increased, this procedure became much more competitive with the other procedures.

The Bounded Wegstein Method also worked very well for all these values of A, and for problems requiring only moderate evaluations times, it would probably have been the choice.

The Method of Successive Substitution again performed increasingly poorly as the value of A was increased.

NON-LINEAR VARIABLES--LINEAR INTERACTION



FUNCTION EVALUATIONS

figure 53

NON-LINEAR VARIABLE--LINEAR INTERACTION

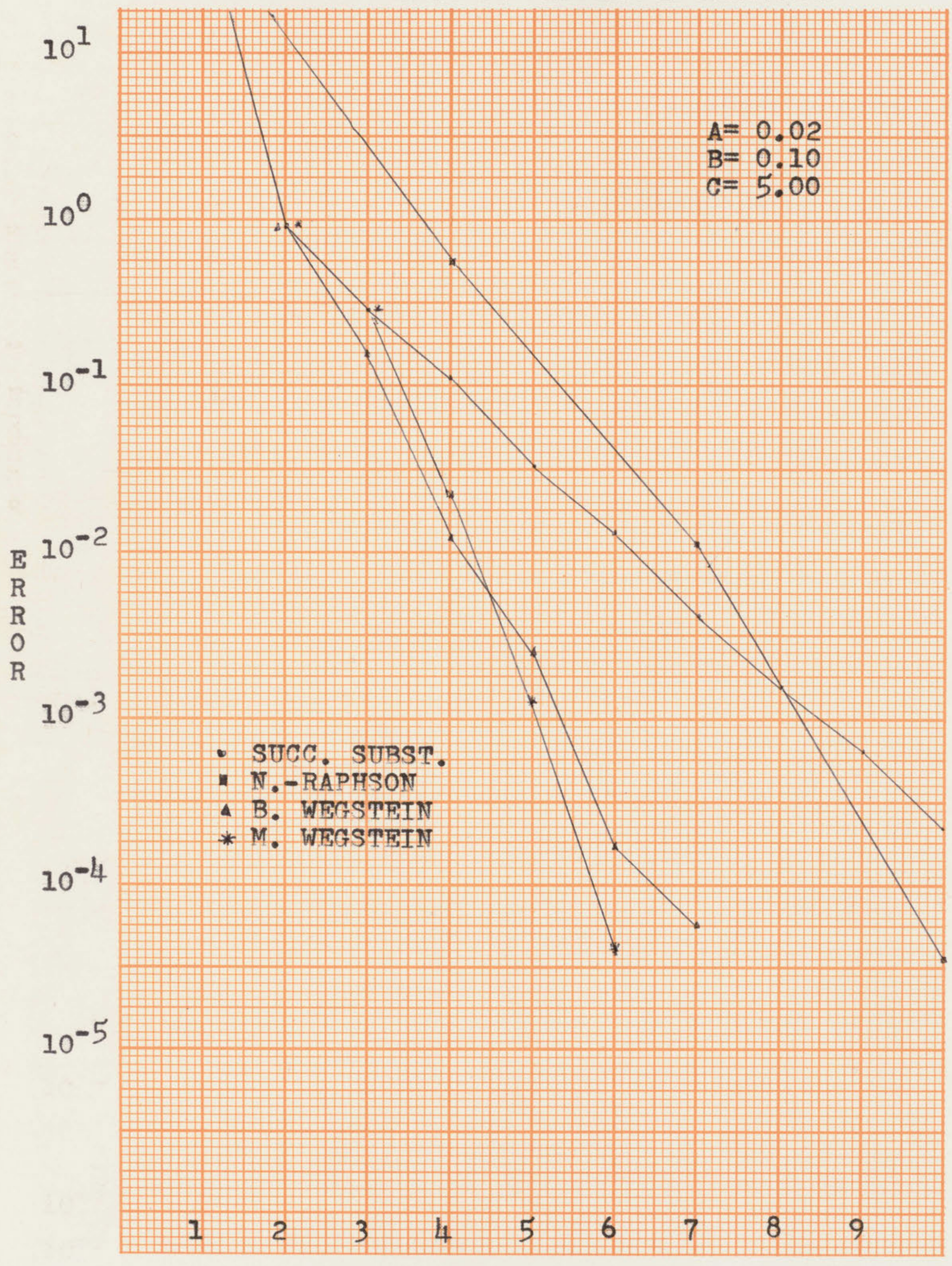


figure 54

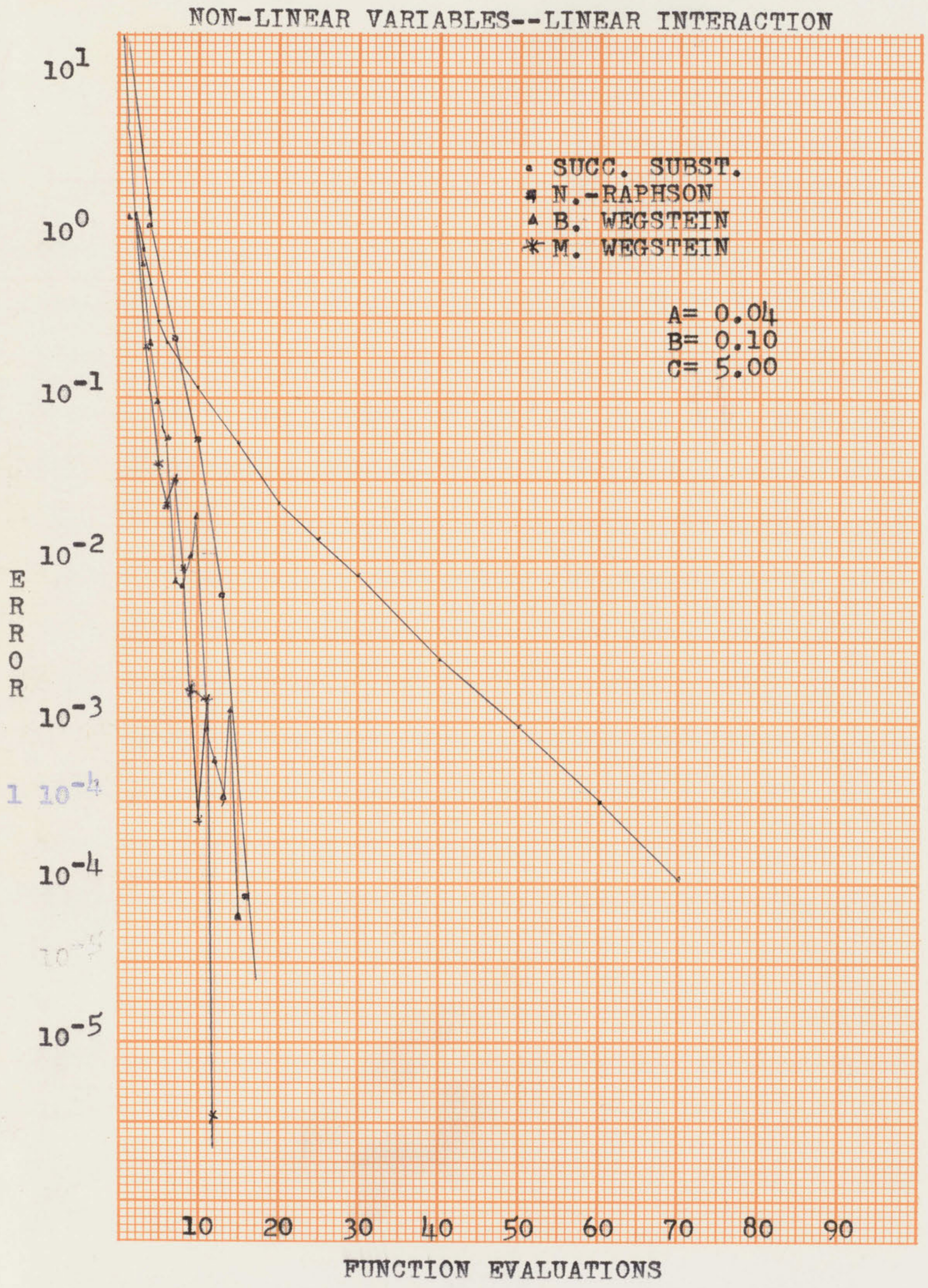

NON-LINEAR VARIABLES--LINEAR INTERACTION
10^1
10^0
10^-1
10^-2
10^-3
10^-4
10^-5
ERROR
• SUCC. SUBST.
■ N.-RAPHSON
▲ B. WEGSTEIN
✳ M. WEGSTEIN
A= 0.04
B= 0.10
C= 5.00
10
20
30
40
50
60
70
80
90
FUNCTION EVALUATIONS

figure 55

V. CONCLUSION

One of the purposes of this paper was to assert the superiority of the Multidimensional Wegstein method over other iterative techniques. For the five engineering examples studied, the Multidimensional Wegstein method's overall performance was much better than that of any of the other procedures. Similarly, the mathematical examples were all solved very well by the Multidimensional Wegstein method, and the Newton-Raphson method. The Newton-Raphson method, however, requires N function evaluations for each iterative step, so that its performance on relations involving more unknowns would probably be lowered.

Viewing the Mathematical problems overall, we can already begin to categorize the four iterative procedures as to the types of problems they are likely to solve well. The Method of Successive Substitution works best when both the degree of intra-variable interaction and the amount of interaction amongst the variables is very low-- values of A and B small. When the exponent of these relations was increased(set to 2), the method became even more sensitive to these values, and would generally solve a smaller range of values. For exponents less than one on either x_1 or x_2 of the relations tested in Section IV, one would ,therefore expect the Method of Successive would work better. Obviously, for the exponent of x_1 and x_2

equal to zero, this method would obtain the solution in one iteration.

The Newton-Raphson method appears to work very well over all the classes of equations studied. This procedure bases its new guess upon a linearized Taylor series expansion about the present guess. For all these problems, such an approximation would represent the system well, hence the good performance of the procedure. The main drawback of this method, again, is its method of obtaining the derivative matrix. These test equations involved only two variables, so three function evaluations were needed for each iterative step. For systems of N variables, N+1 function evaluations are needed for each new step. I suspect that these extra function evaluations would offset the extra speed exhibited by the procedure for these test problems. The Newton-Raphson method performed rather poorly for the Engineering problems.

The Bounded Wegstein Method appears to solve almost exactly the same range of equations as the method of Successive Substitution. For the equations it does solve, however, it works very well. As the degree of interaction increases, it becomes progressively more erratic, and eventually diverges. This result seems reasonable; for, the Bounded Wegstein method was designed to solve equations involving only one variable, and this situation would be closely approached for small values of B.

The Multidimensional Wegstein method, on the other hand, was designed for systems of several variables; it is a multi-dimensional extension of the reasoning behind Wegstein's method. This procedure solves a much wider range of problems than does the method of Successive Substitution, and the Bounded Wegstein methods. Problems involving little interaction were solved very rapidly, as were problems involving moderately large degrees of interaction. Judging from its performance on the engineering example problems, I suspect that it becomes even more effective on problems involving several variables. Clearly, this iterative method appears quite promising, and should be investigated further.

The second objective of my work was to investigate the pedagogical values of the simple sets of equations employed in Section IV, Mathematical Examples. The broad classifications that I just made based on these Mathematical models allows one to separate particular aspects of equations that affect the rates of convergence attainment. This sort of comparison is not as easily obtained by comparing several engineering problems alone. It is only useful, however, if it can be simply related to the actual engineering examples.

The equations for the Aluminum Purification problem all have exponents of one or less and values for A of 0.0. This problem is solved best by the Multidimensional Wegstein method The method of Successive Substitution works moderately well for all values of the pressure. The Bounded Wegstein, however, only works for pressures greater than one. Increasing the pressure decreases the degree of interaction. This result is in line with our expectations, for the Bounded Wegstein method was very sensitive to the amount of interaction, but problems that it could solve, it solved very rapidly. The Newton-Raphson method required about as many iterations as the Multidimensional Wegstein method; but, because it required six function evaluations for each iteration, it appears much slower than the Multidimensional method.

In the equations for the Combustion problem, A would be zero for all the variables, except x_4, whose exponent is -1. Intra-variable interaction, therefore, would be very small. The values corresponding to B, however, vary from zero to moderately high values. Non-linearities in the variable interactions are complicated, but all correspond to an order less than or equal to one.

The Multidimensional Wegstein method was the quickest method at solving this problem, with the method of Successive Substitution being the only other method to obtain solutions. Apparently the degree of interaction is high enough to cause the Bounded Wegstein method to diverge. The Newton-Raphson method, on the other hand, failed because of a matrix inversion failure. This results from a singularity in the derivative matrix. This singularity may have been prevented if I used "extra" precision in the calculations. I suspect, however, that the complete independence of some of the relations from the rest of the equations in the problem formulation, might have led to the inversion problems.

The rates of convergence attainment for both the methods that solved the problem were not affected by the total pressure of the system. This too is reasonable, for the pressure term only affects the equation for the fourth unknown. This term is damped by a very small coefficient, so that its value never becomes significantly large.

The performance of the methods for this example seem to agree with what one would expect, judging from the Mathematical Examples. Admittedly, these two examples weren't ideally matched to the models, that I have investigated. The Nagiev example problem, however, resembles very much the linear equations with linear interaction found in Section IV. The values of A range between 0.08 and 0.67, with most of them being around 0.5. The interaction terms, B, are moderately sized. Five of them are greater than 0.1, with one of these greater than 1.0. The plots of error versus function evaluations for this example, Fig. 21, greatly resembles Fig. 29, the plot of error versus function evaluations for B=0.4, A=0.5, C=5.0. The Newton-Raphson method again suffered a matrix inversion failure, though others have reported solutions to this problem by this method. Use of double precision would probably eliminate this problem.

Since the correlation between the Mathematical and the Engineering examples appears so good, the Mathematical equations appear to allow one to predict how well procedures will perform on various classes of problems. One possible way to implement this knowledge, is to test each new iterative procedure on these and other Mathematical example problems and compare their performance to that of methods already tested, as I just did. One can then choose which procedure to use for a particular problem by matching the problems characteristics to the various mathematical models, and choosing the iterative procedure from those methods known to work well for its particular characteristics.

VI SUGGESTIONS FOR FURTHER STUDY

I believe this procedure merits further investigation and development. My Mathematical examples tested only two variable equations. It would be interesting to see what the affect of extending this analysis to sets containing several more unknowns would be.(Most engineering problems involve 4-7 variables).

The exponents of the variables and the interaction terms is the next most interesting area to probe. I've investigated the case of the exponents equal to 0, 1, and 2. Engineering problems often involve fractional exponents, so the region between 0 and 1 is very important.

Another type of interaction often encountered is x_1x_2 type interactions. Probably, recognizing this as a second order interaction would suffice, but this needs to ve verified.

The Dominant Eigen Value method and the Complex method would be good methods to next compare the Multidimensional Wegstein method too, for they approach the problems differently from the other methods I programmed. The application of the Multidimensional Wegstein method to several large plant designs would probably be the most convincing evidence of its overall usefullness, however.

VII FOOTNOTES

1. Kwon Y. J. "Assymptotic Convergence Techniques and the Ececutive Concept." University Microfilms Limited, Wycomb, England (1969).

2. Ibid. p. 268

3. Meissner H.P. Processes and Systems in Industrial Chemistry. M. I. T. Cambridge, Massl p. 238.

4. Napthali L. M. Chemical Engineering Progress., 60 (9), 70-74. (1964)

5. Loc. cit., p. 134.

6. Nagiev, M.F., "The theory of Recycle Processes in Chemical Engineering", MacMillian Company, New York (1964)

APPENDIX A

This appendix contains the listings of all the supplemental subprograms used by the various convergence procedures and a description of their functions.

(1) SUBROUTINE ZER9(X,N1,N2). This subroutine places zeroes in each element of the N1 by N2 array, X.

(2) SUBROUTINE PROD9(N1,N2,N3,A,B,C). This Subroutine calculates the product of two vectors $\underline{C} = \underline{A} \cdot \underline{B}$, where $\underline{A}$ is an N1by N2 vector, and $\underline{B}$ is an N2 by N3 vector.

(3) SUBROUTINE INVR9(A,AI,N1,N2). This subroutine inverts the N1 by N2 matrix $\underline{A}$, by performing various manipulations on the matrix until it obtains the identity matrix. The same operations are performed on the identity matrix to obtain the inverse.

(4) SUBROUTINE RITZ9(X,F,N,KOUNT). This subroutine prints the values of $\underline{x}$ and $\underline{F}$, and the iteration count when it is called. The various convergence routines call RITZ9 when they are called with the value of Job=3.

(5) SUBROUTINE XQT9(IDOO,RERR,ABER). This routine initializes the first guesses for the problem, then calls each of the iterative proceduresin turn to solve the simulation.

(1)

```
// FOR
        SUBROUTINE ZER9(X,N1,N2)
C****   THIS SUBROUTINE ZEROES AN N1xN2 MATRIX
        DIMENSION X(10,10)
        DO 100 I=1,N1
        DO 100J= 1,N2
        X(I,J)= 0.0
 100    CONTINUE
        RETURN
        END
```

(2)

```
// FOR
        SUBROUTINE PROD9(N1,N2,N3,A,B,C)
C***    THIS SUBROUTINE CALCULATES THE PRODUCT OF TWO MATRICES
C                           C= A*B
        DIMENSION A(10;10),B(10,10),C(10,10)
        CALL ZER9(C;10,10)
        DO 1000 I=1;N1
        DO 1000 J=1,N3
        DO 5000 K= 1,N2
        C(I,J)= C(I,J)+ A(I,K)*B(K,J)
5000    CONTINUE
1000    CONTINUE
        RETURN
        END
```

(3)

```
// FOR
        SUBROUTINE INVR9(A,AI,N1,N2)
C****   THIS SUB INVERTS MATRIX A AND RETURNS IT IN MATRIX AI
        DIMENSION A(10,10),B(10,10),C(10,10),AI(10,10)
C****   ESTABLISH THE IDENTITY MATRIX AND SUBPLANT A IN B
        CALL ZER9(AI,10,10)
       DO 100 I= 1,N1
        AI(I,I)= 1;0
        DO 100 J=1,N2
        B(I,J)=A(I,J)
100     CONTINUE
        JK= 1
        JKP= 2
```

(continued)

```
C****   THIS SECTION OBTAINS A NONZERO IN THE DIAGONAL
125     CONTINUE
        IF(B(JK,JK)) 200,150,200
150     CONTINUE
        DO 175 KL=1,N2
        XCHG= B(JK,KL)
        B(JK,KL)= B(JKP,KL)
        B(JKP,KL)= XCHG
        XCHGI= AI(JK,KL)
        AI(JK,KL)= AI(JKP,KL)
        AI(JKP,KL)= XCHGI
175     CONTINUE
        JKP=JKP+ 1
        IF(JKP-N1) 125,125,6969
200     CONTINUE
C****   OBTAIN A ONE IN THE DIAGONAL
        DIV= B(JK,JK)
        DO 250 J= 1,N2
        B(JK,J)= B(JK,J)/DIV
        AI(JK,J)= AI(JK,J)/DIV
250     CONTINUE
C****   OBTAIN ZEROES IN REST OF COLUMN
        DO 300 I= 1,N1
        DIV= B(I,JK)
        IF(I-JK) 260,300,260
260     CONTINUE
        DO 300 J= 1,N2
        B(I,J)= B(JK,J)*DIV-B(I,J)
        AI(I,J)= AI(JK,J)*DIV- AI(I,J)
300     CONTINUE
C****   GO ON TO NEXT ROW
        JK=JK+ 1
        JKP= JK+ 1
        IF(JK-N1) 125,125,330
6969    CONTINUE
        WRITE(3,9000)
9000    FORMAT(' MATRIX INVERSION UNRESOLVED')
        CALL EXIT
330     CONTINUE
        DO 400 JK= 1,N2
        IF(B(JK,JK)) 350,6969,400
350     CONTINUE
        DO 400 J=1,N2
        AI(JK,J)= -AI(JK,J)
400     CONTINUE
        RETURN
        END
```

(4)
```
// FOR
        SUBROUTINE RITZ9(X,F,N,KOUNT)
C****   THIS ROUTINE PRINTS VALUES OF ITERATIONS
        DIMENSION X(10),F(10),ERR(10)
        DO 100 I= 1,N
        ERR(I)= ABS(F(I)-X(I))
        IF(ABS(ERR(I)/X(I))-ERR(I)) 100,100,75
75      ERR(I)= ABS(ERR(I)/X(I))
100     CONTINUE
        WRITE(3,9400) KOUNT,(X(I), I=1,N)
9400    FORMAT(//,' ITER=',I4,' X=',5(E10.3,5X))
        WRITE(3,9500)(ERR(I),I=1,N)
9500    FORMAT(' ERROR=',7X,5(E10.3,5X))
        RETURN
        END
```

(5)
```
// FOR
        SUBROUTINE XQT9(IDOO,RERR,ABER)
        DIMENSION X(10),F(10),RERR(10),ABER(10)
        GO TO(50,600),IDOO
50      CONTINUE
C****   THIS PROGRAM TESTS THE ITERATIVE PROCEDURES BY
C       CALLING EACH METHOD IN SUCCESSION
        N=2
        MAXIT=100
        JOBB=3
        JOB=0
100     CONTINUE
        CALL FUNV9(X,F,JOBB)
        METH=1
        JOBB=4
        IF(JOBB-10) 150,700,150
150     CONTINUE
        X(1)= 0.0
        X(2)= 0.0
        GO TO (200,300,400,500),METH
200     CONTINUE
        CALL SS9(N,X,F,MAXIT,RERR,ABER,JOB)
        CALL FUNV9(X,F,5)
        GO TO 600
```

(Continued)

```
300     CONTINUE
        CALL BWEG9(N,X,F,MAXIT,RERR,ABER,JOB)
        CALL FUNV9(X,F,5)
        GO TO 600
400     CONTINUE
        CALL NEWR9(N,X,F,MAXIT,RERR,ABER,JOB)
        CALL FUNV9(X,F,5)
        GO TO 600
500     CONTINUE
        CALL MVWG9(N,X,F,MAXIT,RERR,ABER,JOB)
        CALL FUNV9(X,F,5)
600     CONTINUE
        METH= METH+ 1
        IF(METH-4) 150,150,100
700     CONTINUE
        RETURN
        END
```

APPENDIX B ITERATIVE ROUTINES

(1) SUCCESSIVE SUBSTITUTION

```
// FOR
        SUBROUTINE SS9(N,X,F,MAXIT,RERR,ABER,JOB)
C****   THIS SUB SOLVES SYSTEM OF EQTS BY SUCC. SUBST.
        DIMENSION X(10),F(10),RERR(1o),ABER(10)
        KOUNT= 1
100     CONTINUE
        CALL FUNV9(X,F,1)
C****   TEST CURRENT SOLUTION FOR CONVG.
        DO 150 K=1,N
        ERROR= ABS(F(K)-X(K))
        IF(ERROR-RERR(K)*ABS(F(K))-ABER(K)) 150,150,200
150     CONTINUE
C****   SUCCESSFUL CONVERGENCE, RETURN TO CP
        JOB= 1
        CALL RITZ9(X,F,N,KOUNT)
        WRITE(3,9100) KOUNT
9100    FORMAT(' CONVG OBTAINED IN',I4,' ITERS BY THE METHOD
       1 OF SUCC SUBST')
        RETURN
200     CONTINUE
        IF(JOB-3) 240,220,240
220     CONTINUE
        CALL RITZ9(X,F,N,KOUNT)
240     CONTINUE
C****   NOT YER CONVG. TEST FOR MAX NO OF ITS.
        IF(KOUNT-MAXIT) 250,300,300
250     KOUNT= KOUNT+ 1
        DO 275 K=1,N
        X(K)= F(K)
275     CONTINUE
        GO TO 100
C****   UNSUCCESSFUL CONVG. RETURN TO CP
300     CONTINUE
        CALL RITZ9(X,F,N,KOUNT)
        WRITE(3,9200)
9200    FORMAT(' MAX. NO OF ITS EXCEEDED, NO CONVG. BY SUCC SUB.')
        JOB= 2
        RETURN
        END
```

(2)

```
// FOR
        SUBROUTINE BWEG9(N,X,F,MAXIT,RERR,ABER,JOB)
C****   THIS ROUTINE CALCULATES RECYCLE PROBLEMS UNSIN THE
C       BOUNDED WEGSTEIN METHOD ON EACH VARIABLE SEPARATELY
        DIMENSION X(10),F(10),RERR(10),ABER(10),XOLD(10),FXOLD(10)
        JOBB=1
        KOUNT= 1
        QMIN= -20
C****   FIRST ITERATION BY SUCC. SUBST.
        CALL FUNV9(X,F,JOBB)
        DO 100 I=1,N
        XOLD(I)= X(I)
        FXOLD(I)= F(I)
        X(I)= F(I)
100     CONTINUE
125     CONTINUE
        CALL FUNV9(X,F,JOBB)
        IF(JOB-3) 140,130,140
130     CONTINUE
        CALL RITZ9(X,F,N,KOUNT)
140     CONTINUE
        IF(KOUNT-MAXIT) 200,200,150
150     CONTINUE
        JOB= 2
        CALL RITZ9(X,F,N,KOUNT)
        WRITE(3,9000)
9000    FORMAT(' MAX NUMBER OF ITS EXCEEDED, NO CONVG')
        RETURN
200     CONTINUE
C****   INCREMENT ITERATION COUNT
        KOUNT=KOUNT+ 1
C****   TEST FOR CONVERGENCE
        DO 300 I=1,N
        IF(ABS(F(I)-X(I))-RERR(I)*ABS(X(I))-ABER(I)) 300,300,400
300     CONTINUE
C****   IF DO LOOP COMPLETED, CONVG. COMPLETED
        CALL RITZ9(X,F,N,KOUNT)
        WRITE(3,9100) KOUNT
9100    FORMAT(' CONVG. ACHIEVED IN',I8,' ITERATIONS')
        JOB=1
        RETURN
400     CONTINUE
```

(Continued)

```
C****   CALCULATE CANVG. ACCEL PARAM, THEN X
        DO 600 I=1,N
        W= (X(I)-XOLD(I))/(F(I)-FXOLD(I))
        Q= 1.0/(1.0-W)
C****   LIMIT Q BETWEEN QMIN AND 0.0
        IF(Q-QMIN) 425,450,450
425     Q=QMIN
450     CONTINUE
        IF(Q) 550,550,525
525     Q=0.0
550     CONTINUE
C****   PREPARE FOR THE NEXT ITERATION
        XOLD(I)= X(I)
        FXOLD(I)= F(I)
        X(I)= Q*XOLD(I)+ (1.0-Q)*FXOLD(I)
600     CONTINUE
        GO TO 125
        END
```

(3)

```
// FOR
        SUBROUTINE NEWR9(N,X,F,MAXIT,RERR,ABER,JOB)
C****   CONVGS. RECYCLE PROBLEMS USING THE NEWTON-RAPHSO METH .
C       DERIVS. OBTAINED BY SMALL PERTURB. ABOUT VARS.
        DIMENSION X(10),F(10),DVFX(10,10),RERR(10),ABER(10)
        DIMENSION FPT(10),XPT(10),DVFXI(10,10)
        KOUNT= 0
C****   OBTAIN THE CURRENT FUNCTION VALUES
        CALL FUNV9(X,F,2)
        IF(JOB-3) 75,50,75
50      CONTINUE
        CALL RITZ9(X,F,N,KOUNT)
75      CONTINUE
C*****  STORE THE POINTS IN FPT AND XPT
100     CONTINUE
        KOUNT=KOUNT+ 1
        DO 150 K= 1,N
        XPT(K)= X(K)
        FPT(K)= F(K)
150     CONTINUE
```

(Continued)

```
C****    FILL IN THE DERIVATIVE MATRIX
         DO 300 I= 1,N
         X(I)= XPT(I)+ .00001*XPT(I)+ .0001
         DELX= .00001*XPT(i)+ .0001
         CALL FUNV9(X,F,2)
         DO 250 J= 1,N
         DVFX(J,I)= (F(J)-FPT(J))/DELX
250      CONTINUE
         X(I)= XPT(I)
300      CONTINUE
C****    INVERT THE JACOBIAN MATRIX
         CALL INVR9(DVFX,DVFXI,N,N)
C****    COMPUTE THE NEW APPROX TO THE SOLUTION
         DO 500 I=1,N
         DELX= 0.0
         DO 400 J=1,N
400      DELX=DVFXI(I,J)*FPT(J)+ DELX
500      X(I)= X(I)-DELX
         CALL FUNV9(X,D,2)
         IF(JOB-3) 550,525,550
525      CONTINUE
         CALL RITZ9(X,F,N,KOUNT)
550      CONTINUE
C*****   TEST CURRENT SOL'N FOR CONVG
         DO 600 K=1,N
         IF(ABS(F(K))-RERR(K)*ABS(X(K))-ABER(K)) 600,600,800
600      CONTINUE
C****    RETURN TO CALLING PROGRAM, SUCCESSFUL CONVG.
         JOB= 1
         CALL RITZ9(X,F,N,KOUNT)
         WRITE(3,9100) KOUNT
9100     FORMAT(' SOLN IN',I4,' ITERS BY NEWR METHOD')
         RETURN
C****    BEGIN NEW ITERATION
800      IF(KOUNT-MAXIT) 100,900,900
900      CONTINUE
         CALL RITZ9(X,F,N,KOUNT)
         WRITE(3,9000)
         JOB =2
9000     FORMAT(' CONVG TERMINATED, MAX ITER. CNT EXCEEDED')
         RETURN
         END
```

(4)

```
// FOR
      SUBROUTINE MVWG9(N,X,F,MAXIT, RERR,ABER,JOB)
C****  THIS ROUTINE SOLVES RECYCLE PROBS USING THE MULTI-
C      DIMENSIONAL WEGSTEIN METHOD
       DIMENSION X(10),F(10),RERR(10),ABER(10),XT(10,10),
       DIMENSION FT(10,10),DXI(10,10),A(10,10),DX(10,10)
       DIMENSION DF(10,10),XNEW(10),XNEW2(10)
       JOBB= 1
       KOUNT= 0
C****  FIRST OBTAIN N SOLUTIONS BY SUCCESSIVE SUBST.
C      NEWEST FURTHEREST TO THE RIGHT
       CALL FUNV9(X,F,JOBB)
       DO 200 K=1,N
       DO 100 I= 1,N
       XT(I,K)= X(I)
       FT(I,K)= F(I)
       X(I)= F(I)
100    CONTINUE
       CALL FUNV9(X,F,JOBB)
       KOUNT= KOUNT+ 1
       IF(JOB-3)200,150,200
150    CONTINUE
       CALL RITZ9(X,F,N,KOUNT)
200    CONTINUE
C****  TEST FOR CONVG BET NEWEST X,F,PAIR
250    CONTINUE
       DO 300 K=1,N
       IF(ABS(F(K)-X(K))-RERR(K)*ABS(X(K))-ABER(K)) 300,300,325
300    CONTINUE
C****  RETURN TO CALLIN PG., SUCCESS CONVG.
       JOB =1
       CALL RITZ9(X,F,N,KOUNT)
       WRITE(3,9100) KOUNT
9100   FORMAT(' CONVG IN ',I4,' ITS BY THE MULTI WEG METHOD')
       RETURN
C****  BEGIN NEW ITERATION
325    CONTINUE
       IF(KOUNT-MAXIT) 400,350,350
```

(Continued)

```
350       CONTINUE
          CALL RITZ9(X,F,N,KOUNT)
          WRITE(3,9000)
9000      FORMAT(' CONVG. TERMINATED, MAX ITS EXCEEDED')
          JOB=2
          RETURN
400       CONTINUE
C****     CALCULATE DF AND DX
          DO 500 J=1,N
          DO 600 I=1,N
          DX(I,J)= XT(I,J)-X(I)
          DF(I,J)= FT(I,J)-F(I)
600       CONTINUE
500       CONTINUE
C****     COMPUTE A MATRIX
          CALL INVR9(DX,DXI,N,N)
          CALL PROD9(N,N,N,DF,DXI,A)
C****     COMPUTE Q
          DO 650I=1,N
          DO 625 J=1,N
          DX(I,J) = A(I,J)
625       CONTINUE
          DX(I,I)=DX(I,I)-1.0
650       CONTINUE
C* * * * * DX NOW CONTAINS A-I
          CALL INVR9(DX,DF,N,N)
          CALL PROD9(N,N,N,DF,A,DXI)
C* * * * * DXI NOW CONTAINS Q
C****     CALCULATE FIRST PART OF X**N+1
          DO 670 I= 1,N
          XNEW(I)=0.0
          DO 670 K=1,N
          XNEW(I)= XNEW(I)+ X(K)*DXI(I,K)
670       CONTINUE
C****     CALCULATE (I-Q)
          DO 800 I= 1,N
          DO 775 J=1,N
          DX(I,J)= -1.0*DXI(I,J)
775       CONTINUE
          DX(I,I)=DX(I,I)+ 1.0
800       CONTINUE
```

(Continued)

```
C*****+  CALCULATE SECOND PART OF X**N+1
         DO 825 I=1,N
         XNEW2(I)= 0.0
         DO 825 K=1,N
         XNEW2(I)= XNEW2(I)+ F(K)*DX(I,K)
825      CONTINUE
C****    MOVE ALL PTS OVER TO PREPARE FOR NEXT ITERATION
         DO 880 J= 1,N
         K= J+1
         DO 850 I=1,n
         XT(I,J)= XT(I,K)
         FT(I,J)= FT(I,K)
850      CONTINUE
880      CONTINUE
         DO 750 I= 1,N
         XT(I,N)= X(I)
         FT(I,N)= F(I)
750      CONTINUE
C****    SUM THE PARTS
         DO 900 J= 1,N
         X(J)= XNEW(J)+ XNEW2(J)
900      CONTINUE
C****    INCREMENT ITER COUNT
         KOUNT= KOUNT+1
         CALL FUNV9(X,F,JOBB)
         IF(JOB-3) 250,950,250
950      CONTINUE
         CALL RITZ9(X,F,N,KOUNT)
         GO TO 250
         END
```