12-2-1971

Computer Solution to Inverse Problems of Elliptic Form: $V^2U(x,y) = g(a,U,x,y)$

Frederick Alvin Jeter
Portland State University

AN ABSTRACT OF THE THESIS OF Frederick Alvin Jeter for the Master of Science in Applied Science presented December 2, 1971.

Title: Computer Solution to Inverse Problems of Elliptic Form: $\nabla^2 U(x,y) - g(a,U,x,y) = 0$

APPROVED BY MEMBERS OF THE THESIS COMMITTEE:

[redacted]

R.W. Rempfer, Chairman

[redacted]

George Lendaris

[redacted]

Phillip Gold

[redacted]

John Casti

One important aspect of our present age of monolithic high speed computers is the computer's capability to solve complex problems hitherto impossible to tackle due to their complexity. This paper explains how to use a digital computer to solve a specific type of problem; specifically, to find the inverse solution of a in the elliptical equation $\nabla^2 U(x,y) = g(a,U,x,y)$, with appropriate boundary conditions. This equation is very useful in the electronics field. The knowns are the complete set of boundary values of U(x,y) and a set of observations taken on internal points of U(x,y).

Given this information, plus the specific form of the governing equation, we can solve for the unknown $\underline{a}$.

Once the computer program has been written using the technique of quasilinearization, Newton's convergence method, discrete invariant imbedding, and the use of sensitivity functions, then we take data from the computer results and analyse it for proper convergence. This data shows that there are definite limits to the usefulness and capability of the technique.

One of the results of this study is the observation that it is important to the proper functioning of this problem solving technique that the observations taken on $U(x,y)$ are placed in the most efficient locations with the most efficient geometry in the region of largest effectiveness. Another result deals with the number of observation points used: too few gives insufficient information for proper program functioning, and too many tends to saturate the effectiveness of the observations.

Thus this paper has two objectives. First to develop the technique, and secondly to analyse the results from the realization of the technique through the use of a computer.

COMPUTER SOLUTION TO INVERSE PROBLEMS OF ELLIPTIC FORM: $\nabla^2 U(x,y) = g(a,U,x,y)$

by

FREDERICK ALVIN JETER

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

APPLIED SCIENCE

Portland State University
1971

TO THE OFFICE OF GRADUATE STUDIES:

The members of the committee approve the thesis of Frederick Alvin Jeter presented December 2, 1971.

[redacted]

R.W. Rempfer, Chairman

[redacted]

George Mendaris

[redacted]

John Casti

[redacted]

Phillip Gold

APPROVED:

[redacted]

Nan-teh Hsu, Head, Dept of Applied Science

[redacted]

David T. Clark, Dean of Graduate Studies

TABLE OF CONTENTS

LIST OF FIGURES

PREFACE

The solution of elliptical equations has great impact upon engineering fields. For example, in magneto-hydrodynamics we have this problem to solve

$$(1)\ \nabla^2 U(x,y) - a_1 \exp(a_2 U(x,y)) = 0 .$$

In the electronics field we have Poisson's Equation

$$(2)\ \nabla^2 U(x,y) + \rho(U,x,y)/\epsilon = 0 .$$

This paper was born out of a long-term study of cathode ray tubes and their functioning. One equation, which is the example given in this paper, is an approximation to Poisson's Equation for certain regions of electric fields $U(x,y)$ and charge distribution $\rho(U,x,y)/\epsilon$ where

$\rho(U,x,y)/\epsilon \approx aU(x,y)$, thus

$$(3)\ \nabla^2 U(x,y) - aU(x,y) = 0 .$$

Thus, in engineering, we want to be able to solve elliptic problems of the general type

$$(4)\ \nabla^2 U(x,y) - g(a,U,x,y) = 0 ,$$

where possible forms of g are shown in Eqs (1) through (3) above.

This paper will develop some problem solving techniques for the general case, Eqn (4). In particular, it explains the computer program written to solve for the constant $\underline{a}$ in Eqn (3), and an analysis of the computer results to establish the validity of the method.

Again, the specific problem form is the general elliptical equation, Eqn (4), where $U(x,y)$ is contained in the function space $C^2(R^2)$, where $x,y \in R^1$, $\underline{a}$ in general is a real vector of unknown quantities, and $U(x,y) = f(x,y)$ are the known boundary values on the entire boundary of our region of definition $R \subset R^2$. The function g will be assumed sufficiently smooth to ensure a unique, continuous solution to Eqn (4) in R.

The specific example which will be dealt with in detail is called "linex" and is of the form of Eqn (3). Here we use the specific method of this paper to solve for, in this simple case, the one-dimensional unknown $\underline{a}$. In this situation we have the analytic solution $\underline{a} = \nabla^2 U(x,y)/U(x,y)$. So, for any point in R an analytic solution for $\underline{a}$ can be readily found. But there are other specific problems, like Eqn (1) where there is no quick way of solving for the

unknown $\underline{a} = (a_1, a_2)$ since $\underline{a}$ is a vector in this example.

The initial development of this solution method is derived from work done at the University of Sou thern California by Dr. Edward Angel. The general theoretical part can be found in his paper [1]. My main contribution to this work lies in the analysis of the results.

The Table of Contents for this work gives a fairly accurate outline on the contents. The order of the chapters is harmonious with the natural succession of first working out the technique and second trying out the theory on a computer.

As will become evident in the Introduction, the inverse nature of solution means that, for example, in Eqn (3) we want to solve for $\underline{a}$ but first it is necessary to find a solution for U(x,y) with a specific guess for $\underline{a}$. The inverseness lies in the method of solution necessary to find U(x,y).

CHAPTER I

INTRODUCTION

In many problems dealing with the real world, one finds a common situation where the governing equations of the phenomenon are known except for one parameter, scalar or vector [3]. Since there are many more <u>forms</u> of problems in need of solutions, one must limit his scope of pursuit to a small, small subset of the entire problem spectrum. We shall limit ourselves to the specific task of solving Inverse Boundary-Value Problems of the general elliptic type,

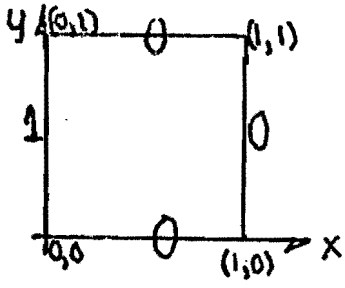
$$(1) \quad \nabla^2 U(x,y) - g(x,y,U,\underline{a}) = 0,$$

with $U(x,y) = f(x,y)$ defining the boundary values. We shall examine the specific results of using an IBM 1130 computer for the problem solution. In this general form, $\underline{a}$ is the only unknown, since $U(x,y)$ can be determined because we know the boundary conditions f and the region R.

Our investigation will mainly involve a linear example with $\underline{a}$ as a scalar:

$$(2) \quad \nabla^2 U(x,y) - aU(x,y) = 0,$$

with U = 1 along (0,0) to (0,1) and on the other three sides of the unit square, U = 0.



Region of Interest in our Example, LINEX

In order to ascertain the value of $\underline{a}$, for the method of solution used herein, we must have accurate observed knowledge of U at L distinct points, each point specified by its (x,y) coordinates. As we shall discover, the number and placement of the observation points is of utmost importance. Luckily there are rules of thumb as we shall see.

Possible physical examples for our studies might be the non-linear magneto-hydrodynamic problem described by

$$(4) \quad \nabla^2 U(x,y) = a_1 \exp(a_2 U(x,y)),$$

where $\underline{a}$ is a two-dimensional real vector $\underline{a} = (a_1, a_2)$. A simpler version of this same problem would be

$$(5) \quad \nabla^2 U(x,y) = \exp(aU(x,y)),$$

normalized so that $\underline{a}$ is one-dimensional.

See reference [2] for further details of this problem.

There are three important mathematical tools which we shall need to solve for a in Eqn (1):

1) Invariant Imbedding

2) Newton's Method

3) Quasilinearization (if g is a nonlinear function of U)

Quasilinearization is used to translate the nonlinear equation into a sequence of linear problems convergent to the solution of Eqn (1). Given the nonlinear equation:

(6) $\nabla^2 U(x,y) - a_1 \exp(a_2 U(x,y))$, $U=1$ on boundary,

the linear version of the same equation in the unknown function U_{m+1} is:

(7) $\nabla^2 U_{m+1}(x,y) = a_1 \exp(a_2 U_m) + (U_{m+1} - U_m) a_1 a_2 \exp(a_2 U_m)$

$U_{m+1} = 1$ on R.

CHAPTER II

METHOD OF INVERSE SOLUTIONS

In inverse problem solving, we want to find the unknown in a specific governing equation such that the observations made on the process match up with the computer solution of the governing equation with the correct $\underline{a}$-value. We shall employ a method of solution which relies upon the quantity known as the Relative Least Squares Error (R.L.S.E.)

$$(8) \quad S(a) = \sum_{\ell=1}^{L} (U_n(\ell) - RHO(\ell))^2 / RHO^2(\ell),$$

where $U_n(\ell)$ is the value of U derived from using the nth approximation to $\underline{a}$, and $RHO(\ell)$ is the value taken from the ℓth observation data. It is shown in reference [1] that the least squares error method can yeild the correct or proper $\underline{a}$, and in this paper we accept the method's validity, use it, and observe the results.

Taking the minimum over $\underline{a}$ of the least squares error and rewriting,

$$(9) \quad \min_{a} S(a) = \min_{a} \sum_{\ell=1}^{L} (U_n(\ell) - RHO(\ell))^2 / RHO^2(\ell)$$

We can replace (9) by noting that $\underline{a}$ is the solution of

(10) $\nabla S(a) = 0$ OR $\partial S/\partial a_i = S_{a_i}(a) = 0$, $i=1,2,....p.$

Thus $\underline{a}$ is the simultaneous solution of p nonlinear differential equations, whose solution presents quite a formidable computational task in itself.

Apparent at this juncture is the great difficulty in applying the classical gradient method in (10). Gradient methods, in general, require an enormous amount of computing because it is necessary to solve all p equations for $\underline{a}$. Usually many guesses at $\underline{a}$ are necessary. Also gradient solution requires the simultaneous solution of Eqn (1), a very time consuming job. We need another method, one which will be economical on the computer.

CHAPTER III

NEWTON'S CONVERGENCE METHOD

If we regard the original general governing equation,

$$(1)\quad \nabla^2 U(x,y) - g(U,x,y,a) = 0$$
$$U(x,y) = f(x,y) \text{ on boundary}$$

as our equation of interest, we can employ the general method of solution known as Newton's method.

Assuming that the gradient $\nabla S(a)$ is continous in $\underline{a}$, we can utilize Newton's Method as follows: let $\underline{a}_0$ be an initial guess at $\underline{a}$, then Newton's method says that a new and more accurate approximation can be found by:

$$(11)\quad \underline{a}_{k+1} = \underline{a}_k - J(\underline{a}_k)^{-1} \quad \nabla S(\underline{a}_k) \quad \text{see [4]}$$

where $J(a_k)$ is the Jacobian of the system. Namely, J is the matrix whose i,j component is

$$(12)\quad J(\underline{a})_{ij} = S_{\underline{a}_i \underline{a}_j} = \partial^2 S(\underline{a}) / \partial \underline{a}_i \partial \underline{a}_j, \quad i,j=1,2,\ldots,n.$$

If we are near the correct $\underline{a}$, then the convergence rate should be quadratic:

$$(13)\quad \| \underline{a}_{optimal} - \underline{a}_{k+1} \| = O(\| \underline{a}_{optimal} - \underline{a}_k \|^2).$$

Thus, at each interation the number of significant digits

should be approximately doubled if we start with a good a_o. The big problem at this point is the correct evaluation of $\nabla S(\underline{a})$ and $J^{-1}(\underline{a})$.

CHAPTER IV

USE OF SENSITIVITY FUNCTIONS

Like so many jig-saw puzzles, we are finally approaching the point where all the pieces fit in place. Let us take the first and second derivatives of the sensitivity function S(a). Since $\nabla S(\underline{a}_k) = S_{a_i}(\underline{a}_k)$ and

$$J(\underline{a}_k) = S_{a_i a_j}(\underline{a}_k).$$

Now all the parts fit the jigsaw puzzle.

Taking the necessary derivatives of the R.L.S.E. function S(a), we obtain

$$(14)\quad \partial S/\partial a_i = S_{a_i}(\underline{a}) = 2\sum_{l=1}^{L} ((U(L)-RHO(1))\ U_{a_i}(1))/RHO(1)^2$$

$$(15)\quad \partial^2 S/\partial a_i \partial a_j = 2\sum_{l=1}^{L} [(U(1) - RHO(1))\ U_{a_i a_j}(1) + U_{a_i}(1)\ U_{a_j}(1)]\ /RHO(2)^2.$$

We find that these equations have created what are known as the Sensitivity Functions, which we now define as

$$(16)\quad 1)\quad U(x,y) = U(x,y)$$

$$2)\quad U_{a_i}(x,y) = \partial U(x,y)/\partial a_i = V(x,y)$$

$$3)\quad U_{a_i a_j}(x,y) = \partial^2 U(x,y)/\partial a_i \partial a_j = W(x,y)$$

We have now introduced two more variables which are similar to, and closely related to, U(x,y), namely V and W. Since simplification is our goal, let us now examine the U, V, and W functions.

By repeated differentiation of Eqn (1) and interchanging the order of differentiation, we obtain

(1) $\nabla^2 U + g(U,x,y,a) = 0$, $u=f(x,y)$ on boundary

(17) $\nabla^2 U_{a_i} + g_{a_i} + g_U U_{a_i} = 0$, $U_{a_i} = 0$ on boundary

(18) $\nabla^2 U_{a_i a_j} + g_{a_i a_j} + g_{U_{a_i}} U_{a_i} + g_{U_{a_j}} U_{a_i} + g_{UU} U_{a_i a_j} + g_U U_{a_i a_j} = 0$,
$U_{a_i a_j} = 0$ on boundary.

Note that both Eqns (17) and (18) are also linear and differ from Eqn (1) only in their forcing function. Note that Eqn (1) is the forcing function for (17) while (17) is the forcing function for (18). The power of our method resides in these relationships.

CHAPTER V

QUASILINEARIZATION

From the preceeding chapter, we see that it is desirable to have all equations in linear form. When the governing equation is not linear, we quasilinearize to a sequence of linear problems which converge to the solution of the problem of interest. A nonlinear example is

$$\nabla^2 U(x,y) = \exp(aU(x,y) = g(a,U,x,y).$$

Let V be an approximation of U. By Taylor's series expansion about V, we see that

$$g(U) = g(V) + (U-V)g_U(V) + O((U-V)^2). \tag{19}$$

Substituting Eqn (19) into Eqn (1) yeilds

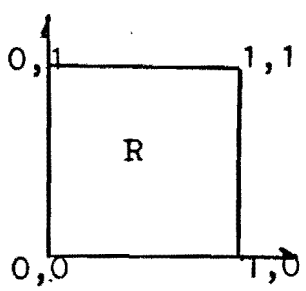
$$\nabla^2 U_{m+1} + g(U_m) + (U_{m+1}-U_m)g_U(U_m) = 0, \tag{20}$$

which is linear in U_{m+1} and assumes the original boundary value. In the limit, $U_m \rightarrow U$ as $m \rightarrow \infty$.

CHAPTER VI

DISCRETE INVARIANT IMBEDDING

We now introduce the needs of our discrete, digital computer. For simplicity of development, we shall assume that the region of interest R is the unit square.



Unit Square: Region of Interest

This assumption is for simplicity of development only, because invariant imbedding is ideally suited for treatment of elliptic equations over very general regions. [1, 5]

Remembering that all governing equations must be linearized, we can write the generalized form of the governing equation as

$$(21) \quad \nabla^2 U(x,y) + H(x,y)U(x,y) = S(x,y)$$

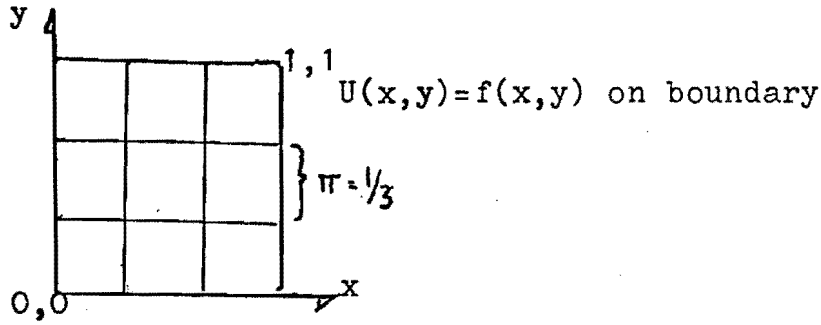
where U represents U, U_{a_i}, $U_{a_i a_j}$, or U_m with appropriate H and S functions.

Since we are developing a computer solution technique,

we now need to translate from the continuous to the discrete. We seek a solution of Eqn (21) at a discrete set of evenly spaced points, (x_i, y_j), forming a grid on the unit square R. We shall choose the points such that the spacing between them is

$$(22)\quad \pi = 1/N, \quad (\underline{\text{not}} \text{ the } \pi=3.1415926\ldots)$$

where N denotes the number of divisions in the grid. For example, if N = 3 we would have the grid



For simplicity, let us define U_{ij} as $U(i\pi, j\pi)$. Applying the standard five point approximation formula for $\nabla^2 U(i\pi, j\pi)$ gives

$$(23)\quad \nabla^2 U_{ij} = \frac{U_{i+1,j} + U_{i,j+1} - 4U_{ij} + U_{i-1,j} + U_{i,j-1}}{\pi^2}$$

Thus, the standard finite difference equation becomes

$$(24)\quad U_{i+1,j} + U_{i,j+1} - 4U_{i,j} + U_{i-1,j} + U_{i,j-1} + \pi^2 H_{i,j} U_{i,j} = \pi^2 S_{i,j},$$

$$i,j = 2,3,\cdots,N-1$$

The boundary values are known from the original continuous equation (1). Now we have $(N-2)^2$ linear algebraic equations to solve.

We shall adopt a vector notation to expedite the solution. Let U_i be the ith column vector of interior points (the boundary points are known from the boundary conditions)

$$(25) \quad U_i = \begin{bmatrix} U_{i2} \\ U_{i3} \\ \vdots \\ U_{iN} \end{bmatrix}$$

let us also define some handy vectors and matrices:

$$(26) \quad Q = (q_{ij}) = \begin{cases} 2 & , i=j \\ -1 & , i-j=1 \\ 0 & , \text{otherwise} \end{cases}$$

$$r_i = (r_{ij}) = \begin{cases} U_{i1} & , j=2 \\ U_{i,N+1} & , j=N \\ 0 & , \text{otherwise} \end{cases}$$

Now we rewrite Eqn (24) using the vector notation

$$(28) \quad U_{i+1} - 2U_i + U_{i-1} - QU_i + r_i + H_iU_i = S_i$$

where

$$(29) \quad H_i = \pi^2 \operatorname{diag}(H_{i2}, H_{i3}, \ldots, H_{iN}), \; i=2,3,\ldots,N$$

$$S_i = \text{column}\,(\pi^2 S_{ij}), \; j=2,3,\ldots,N$$

We now have Eqn (28), a two-point boundary value problem with U_1 and U_{N+1} known and the other boundary conditions entering through the r_i.

Since we are looking for solutions to linear equations, we can seek solutions to Eqn (28) in the form

(30) $U_{i+1} = A_i U_i + b_i$, where A_i are $(N-1)^2$ matrices, b_i are $(N-1)$-dimensional vectors, and the A_i, b_i are <u>independent</u> of U. Here the functional form is recursive, that is, U_2 depends on U_1, U_3 depends on U_2 and so on. To begin the recursive process of solution, we know U_1 from the boundary conditions since U_1 is part of the boundary, so is U_{N+1}. Solving for the coefficients A_i and b_i, we put Eqn (30) into Eqn (28) which results in

(31) $A_{i-1} = (2I + Q - H_i - A_i)^{-1}$, $A_N = 0$

(32) $b_{i-1} = A_{i-1}(r_i + S_i + b_i)$, $b_N = U_{n+1}$

Notice that Eqn (31) is in inverse form.

The most important existence condition lies in the existence of the inverse of the matrix in Eqn (31). Since $2I+Q$ is positive definite and A_N is zero from the boundary conditions, the entire set of A_{i-1} matrices are positive if H_i is negative definite. Thus, an inductive argument shows the existence of the inverses for all i.

The power of invariant imbedding for this problem

lies in recognizing and using the solutions to linear equation (30). Also we prefer to use invariant imbedding rather than standard iterative methods like "sucessive over-relaxation" or "alternating direction implicit" because

1) there are no parameters to choose even when the problem is nonlinear,
2) it works well within irregular regions,
3) the solution of the sensitivity equations is easy because of their similarity to Eqn (1).

From the computational viewpoint, the matrix recurrence equation (31) requires a large part of the total computing time for a given a_k, (k = 1,2,...m), but we need only one set of A-matrices for calculations of U, U_{a_i}, and $U_{a_i a_j}$ (ie, U,V, and W).

CHAPTER VII

COMPUTER PROGRAM, GENERAL OUTLINE

(1) If governing equation is nonlinear, linearize

(2) Guess an $\underline{a}_0$.

(3) Solve recursively for the set of A_i matrices, $i = (N, N-1, N-2, \ldots, 2)$ using Eqn (31).

(4) Solve recursively for the b_i's, using Eqn (32).

(5) Solve for U_{i+1}, $(i=1,2,\ldots N)$ using Eqn 30).

(6) Solve for V_{i+1}, using Eqn (16).

(7) Solve for c_i's using same equation as for b_i's.

(8) Use the same routine to solve for the W_{i+1} as for V_{i+1}; Where $W_{i+1} = A_i W_i + c_i$.

(9) Compute sums $S_{a_i}(\underline{a}_k)$ and $S_{a_i a_j}(\underline{a}_k)$ using results from (6) and (8) above and Eqns (14) and (15).

(10) Compute new a_{k+1} from Eqn (11).

(11) If $|\underline{a}_{k+1} - \underline{a}_k| \underset{\text{(Linex)}}{\geq} .005$, repeat (3) through (10), with $\underline{a}_{k+1}$ replacing $\underline{a}_k$, otherwise stop.

CHAPTER VIII

LINEAR EXAMPLE: "LINEX"

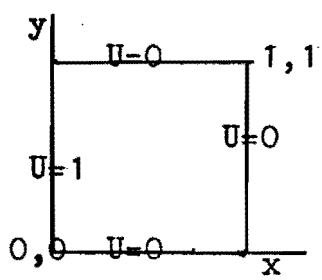
A detailed analysis of computer results from the linear example follows in Chapter XI.

We will examine the equation

(33) $\nabla^2 U(x,y) = aU(x,y)$, where $\underline{a}$ is scaler.

Assume fixed boundary conditions on the unit square with lower left corner being the origin (see figure).

(34) $U(0,y) = 1.0$
$U(1,y) = U(x,0) = U(x,1) = 0$



LINEX Boundary Conditions

We wish to determine $\underline{a}$ such that, from a set of observation points in region R, we can minimize the least squares criterion

(35) $S(a) = \sum_{l=1}^{L} (U(l) - RHO(l))^2/RHO(l)^2.$

The associated sensitivity functions are

(36) $\nabla^2 U_a = aU_a + U$, $U_a = 0$ on boundary,

(37) $\nabla^2 U_{aa} = aU_{aa} + 2U_a$, $U_{aa} = 0$ on boundary,

For simplicity, we write

$$(38)\quad V(x,y) = U_a(x,y)$$

$$W(x,y) = U_{aa}(x,y)$$

Thus, we must solve the set of equations

$$(39)\quad \nabla^2 U - a\,U = 0,$$

$$\nabla^2 V - a\,V = U,$$

$$\nabla^2 W - a\,W = 2V.$$

Note that the V solution depends upon the U solution while the W solution depends upon the V solution.

The discrete version of (39) is

$$(40)\quad U_{i+1} - 2U_i + U_{i-1} - QU_i - \pi^2 a U_i = 0$$

$$(41)\quad \text{" (V replaces U)} = \pi^2 U_i$$

$$(42)\quad \text{" (W replaces U)} = \pi^2 V_i$$

Since in this example $r_i = 0$, we have the following computer program outline:

Computer Program Outline

(1) All equations already linear, thus no quasilinearization necessary,

(2) $A_{i-1} = ((2+\pi^2 a_k)I+Q-A_i)$, $A_N = 0.0$,

(3) $U_{i+1}=A_i U_i$, $U_1 = 1.0$, $(U_N = 0.0)$,

(4) $b_{i-1} = A_{i-1}(b_i + \pi^2 U_i)$, $b_{N-1} = 0$,

(5) $V_{i+1} = A_i V_i + b_i$, $V_1 = 0.0$,

(6) $c_{i-1} = A_{i-1}(c_i + 2\pi^2 V_i)$, $c_{N-1} = 0.0$,

(7) $W_{i+1} = A_i W_i + c_i$, $W_1 = 0.0$,

(8) $a_{k+1} = a_k - (S_a / S_{aa})$,

(9) test $|a_{k+1} - a_k| < .005$, if true exit, if false repeat (2) through (9) with updated estimate of $\underline{a}$.

See Chapter IV to understand where the right-hand side of step (8) comes from.

CHAPTER IX

A NONLINEAR EXAMPLE (WITH NO COMPUTER DATA)

Let us assume that the governing equation is

(43) $\nabla^2 U(x,y) = a \exp(bU(x,y))$ (1), in the unit square,

$U(x,y) = 1$ on boundary.

Linearize Eqn (43)

(44) $\nabla^2 z_{m+1} = a \exp(bz_m) + ab \exp(bz_m)$,

$z_{m+1} = 1$ on boundary,

where

(45) $\lim_{m\to\infty} z_m = U.$

We use repeated differentation of (43) with respect to $\underline{a}$ and $\underline{b}$ (our unknowns) to derive the sensitivity equations:

(46) $\nabla^2 U_a = e^{bU} + abe^{bU}U_a$ (First Derivatives)

(47) $\nabla^2 U_b = aUe^{bU} + abe^{bU}U_b$

(48) $\nabla^2 U_{aa} = 2be^{bU}U_a + abe^{bU}U_{aa} + ab^w e^{bU}U_a^2$

(49) $\nabla^2 U_{bb} = 2\,ae^{bU}U_b + 2abUe^{bU}U_b + aU^2e^{bU} + abe^{bU}U_{bb} + ab^2e^{bU}U_b^2$ (Second Derivatives)

(50) $\nabla^2 U_{ab} = Ue^{bU}U_b + ae^{bU}U_a + abUe^{bU}U_a + abe^{bU}U_{ab} + ab^2e^{bU}U_aU_b$

Boundary conditions are all equal to zero.

(51) $U_a=U_b=U_{aa}=U_{bb}=U_{ab}=0.0$ on boundary of the unit square.

The (normalized) Least Squares Criterion is

$$(52)\quad S(a,b) = \sum_{l=1}^{L}(U(l) - RHO(l))^2/RHO^2(l).$$

Therefore, by differentiation we obtain

$$(53)\quad S_a(a,b) = \sum_{l=1}^{L}2(U(l)-RHO(l))U_a(l)/THO^2(l)$$

$$S_b(a,b) = \text{"(b)}$$

$$S_{aa}(a,b) = \sum_{l=1}^{L}2((U(l)-RHO(l)U_{aa}(l) + U_a^2(l))/RHO^2(l)$$

$$S_{bb}(a,b) = \text{"(b)}$$

$$S_{ab} = S_{ba} = \sum_{l=1}^{L} 2((U(l)-RHO(l)U_{ab}(l) + U_a(l)U_b(l))/RHO^2(l).$$

Newton's method yields

$$(54)\quad a_{k+1} = a_k - (S_{bb}S_a-S_{ab}S_b)/ (S_{aa}S_{bb}-S_{ab}^2),$$

$$b_{k+1} = b_k - (S_{ab}S_a-S_{aa}S_b)/ (S_{aa}S_{bb}-S_{ab}^2).$$

For each value of (a_k,b_k), we solve Eqns (44) and (46) through (54). Since these equations are of similar forms the invariant imbedding procedure is quite efficient. Observations were generated (by Angel) by solving the discretized problem. Angels results for the constants a and b were

Iteration No. k	a_k	b_k
0	1.500	1.500
1	1.682	2.274
2	1.681	2.269
3	1.681	2.269

CHAPTER X

ANGEL'S CONCLUSIONS FROM HIS SPECIFIC PROGRAM [1]

Angel used six observation points but did not specify where in the matrix they are placed or with what type of placement geometry. Also, his computer program was used on a larger computer and he was able to have a 17 x 17 grid while the Portland State University 1130 computer is basically only an 8k machine and the largest matrix that would fit into the computer is an 11 x 11 matrix. (Actually, with a few minor changes a 12 x 12 matrix could be used on the 1130 but the program was already trimmed down to bare minimum, except for programmer comments, etc. which, if taken out, could allow a 12 x 12 matrix to be used.) Most of the data that we will see in the following chapters was computed with a 9 x 9 matrix. Angel used six observations out of a total of 289 points (including boundary). The actual number of points available to Angel was 15 x 15 = 225 points while we had available only 7 x 7 = 49 points.

Angel generated his points by solving the general

equation by series approximation. I used the computer solution with $\underline{a}$ = 2.00000 to "generate" my sets of from 1 to 10 data points. I also tried series solution and the results were very close to the completely discrete solution on the computer.

Runs were made by both Angel and myself using initial guesses of a_o as 1.500 and 4.000. Angel's results (sum total of all data recorded by Angel in his paper) are listed here:

Iteration No.	$(a_r)_1$	$(a_r)_2$
0	1.5000	4.0000
1	2.2399	2.9832
2	2.0568	2.3379
3	2.0403	2.0778
4	2.0401	2.0409
5	2.0401	2.0401
6	------	2.0401

1.9987 -- for $\pi = 1/8$
(similar to my data)

The error in the final $\underline{a}$ is presumably due to truncation error in the equation of the discrete $\nabla^2 U_{ij}$. By using π = 1/8 in place of π = 1/16, we find a final a = 1.9987, a better approximation than with a grid of π = 1/16.

On the next page is a graph of Angel's results with six observation points compared to my data with four and six observation points.

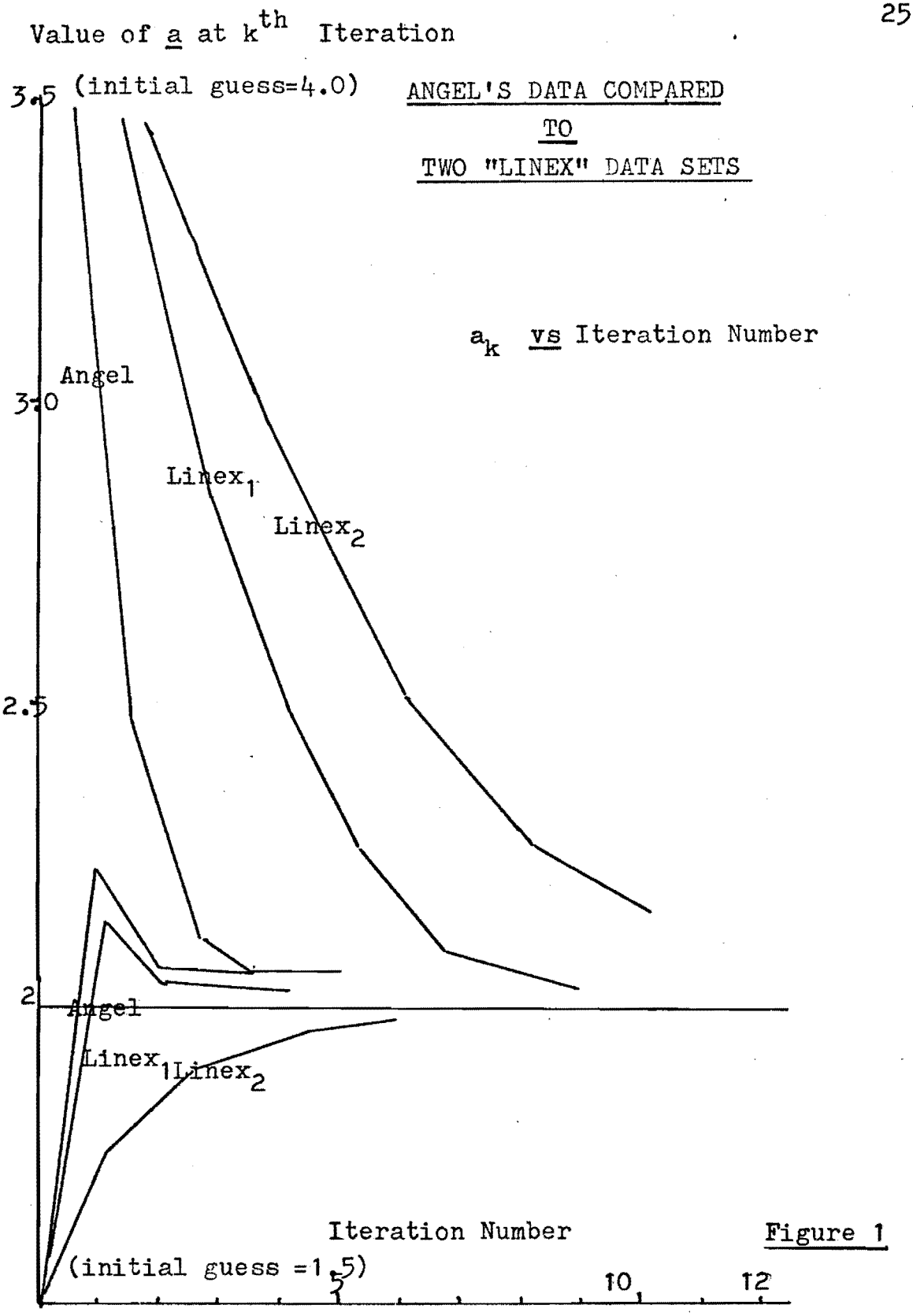

Value of a at k^th Iteration
(initial guess=4.0)
ANGEL'S DATA COMPARED
TO
TWO "LINEX" DATA SETS
a_k vs Iteration Number
3.5
3.0
2.5
2
Angel
Linex1
Linex2
Angel
Linex1
Linex2
Iteration Number
(initial guess =1.5)
5
10
12

Figure 1

CHAPTER XI

"LINEX" INTERPRETATION

This Chapter deals with the interpretation of data from the specific problem which has earlier been called "Linex." In this case, $\underline{a}$ a scaler, is solved for in the linear problem

(55) $\nabla^2 U(x,y) - aU(x,y) = 0$ with boundary conditions

(56) $U(0,y) = 1.00$ and $U(x,0) = U(x,1) = U(1,y) = 0.00$.

The combination of Newton's method and discrete invariant imbedding has, in this example, been shown to be a very effective method of solving this form of simple inverse problem involving a linear elliptic equation with constant coefficients.

The goal has been rapid convergence rate with the only variable being number and placement of observation points. Unlike Angel, I used an explicit solution for $\underline{a} = 2.00$ to extract my observation points rather than a series solution with $\underline{a} = 2.00$.

The data is sorted according to:

1) number of observation points

2) rate of convergence to correct $\underline{a}$

3) symmetry of observation points

The rate of convergence is here defined as the actual (or extrapolated) number of iterations necessary to achieve convergence to an accuracy of .005. If the number is over 13, then it is extrapolated.

Unexpectedly, a quadratic rate of convergence was not seen in all of the test cases. But in the cases where the observation points were apparently in the most effective places and appeared in the right numbers, a rate approximating quadratic convergence is observed. A possible factor, other than observation number and placement, in reducing convergence time might be large errors in the initial guess of $\underline{a}$.

TABLE I

An Approximate Weighting Table of the Number of Interations on a per Data Set.

SYMBOL USED	CLASS DESCRIPTION	NECESSARY ITERATIONS
VF or VG	very fast or very good	<10
F or G	fast or good	$>10 \leqslant 13$
AF or AG	almost fast or good	$>13 \leqslant 16$
P	poor or slow	$>16 \leqslant 25$
VP	very poor or slow	>25
*	Oscil. about correct a but no convergence	initial and continuing in the wrong direction
OVERSHOOT	inital and continuing	overshoot

The class discriptions are approximate and are here used to denote convergence performance between the range of "very fast" to "very slow".

CHAPTER XII

LINEX: SORTING OF DATA SETS

The different classifications of my data sets are chosen to show the success of the specific sets and groups of sets. The order of importance of the set classification are:

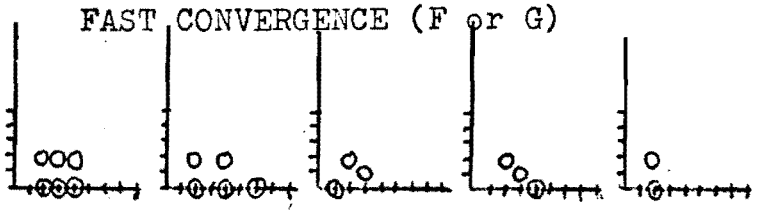
1) Rate of convergence
2) Number of observation points.
3) The shape (symmetry) of the observation points arrangement.

Figure II shows the different data sets arranged according to the classifications of rate as given in Chapter XI.

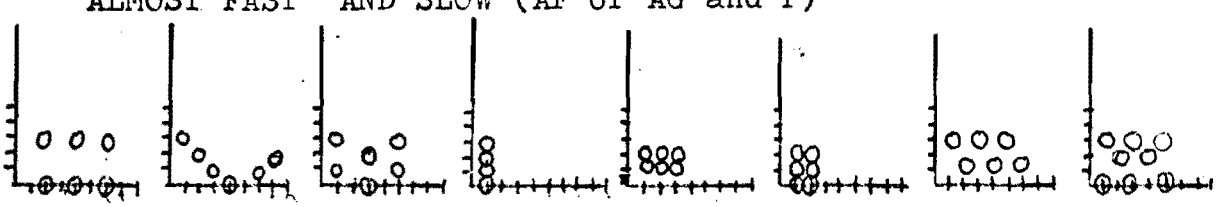
CONFIGURATIONS OF ALL OBSERVATION DATA SETS
CLASSIFIED ACCORDING TO SPEED OF CONVERGENCE

VERY FAST CONVERGENCE (VF or VG)

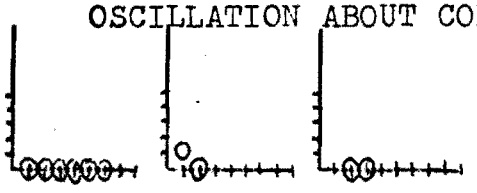
FAST CONVERGENCE (F or G)



ALMOST FAST AND SLOW (AF or AG and P)



OSCILLATION ABOUT CORRECT a VALUE



WRONG DIRECTION OR OVERSHOOT

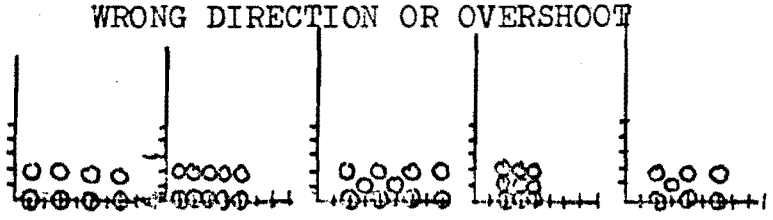


Figure 2

CHAPTER XIII

LINEX: RESULTS OF DATA SORTING

(1) If the initial $\underline{a}_0$ guess is not close enough to the final $\underline{a} = 2.00$, then there is a greater possibility that the final a-guess could approach another local extremum. Theoretically, this cannot occur, at least for linear problems. However, numerical inaccuracies can introduce spurious results.

(2) As the data-arrangement is further from the y-origin symmetry line, the rate of convergence is generally slower.

(3) Data sets arranged symmetrically as in:

```
xoxox
ooooo
xoxox
```

where x is data and o is blank, seem to be the most successful arrangements, especially if they follow (2), above, and have from 2 to 6 observation points.

(4) The method used in solving LINEX is based upon the largest matrix size which could fit in the Portland State University IBM 1130 computer (this being 9 x 9) while Angel's work was done using a 17 x 17 array. This is a ratio of useful non-boundary-value numbers of 49/225 or five times as

many points, giving a matrix ratio (square-root of points ratio) of 2.3 to 1. So there is a very good chance that convergence was slower because of this or, perhaps there is some sort of resonance phenomenon involved.

(5) The most successful data sets seemed to work due to the number of observation points (the successful group all contained 2 to 6 points each). The successful sets are definitely a function of the pattern and placement of observation points.

(6) There are three classifications of the least successful observation point data sets.

a) the sets which were completely off had 7 to 10 observation points/set. One would believe that more data would enhance covergence but such appeared not to be the case,

b) the oscillating data sets all had their observation points on the axis of symmetry (namely, $y=5\pi$). This is definitely significant.

c) for the successful but very slowly convergent sets, see the figure on the next page.

FIGURE OF THE LEVELS OF CONVERGENCE SHOWING THE LOC-ACTION AND RELATIVE FREQUENCY OF THE OBSERVATION POINTS

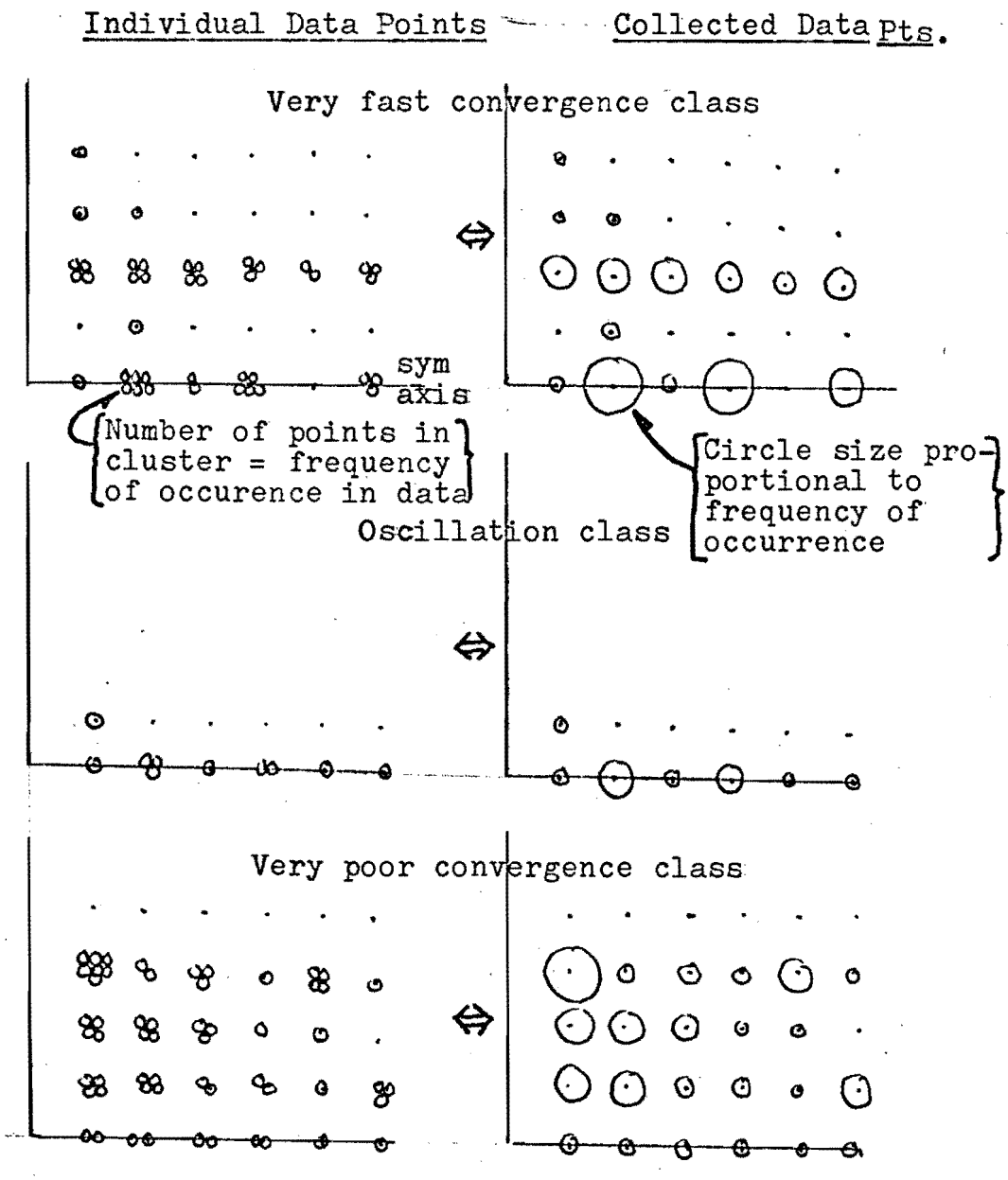


Figure 3

The "Collected Data Pts." show the effects of bunching the "Individual Data Points" into an expression of relative "size" for a given position. ⇔ indicates equivalence.

CHAPTER XIV

CONCLUSIONS

By looking at the data interpretation chapters, it is apparent there is a right way and a wrong way of getting fast convergence from a set of observation points. And, of course, if we can get fast convergence then we have a successful computer solution to the problem at hand.

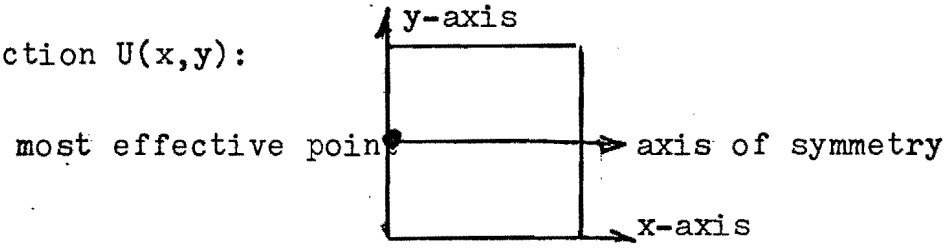
Quite likely there will be other conditions for successful observation points for other geometries. For example, the nonlinear problem discussed in Chapter IX. Here, since there is a boundary value of 1 all around the unit square, successful data points could be anywhere near the boundary not just near the unit value y-axis as in "LINEX."

Returning to LINEX, we obtained fast convergence when the observation data points upon U(x,y) were as near to the origin as possible, while still in an XOXOX formation as shown on page 30. This so-called proper spacing must have something to do with our use of the standard five point approximation formula in discretizing our basic formulas.

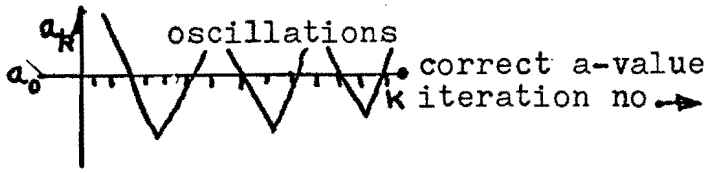
The five point approach has the following symmetry:
oxo
xox .
oxo

Very similar is our most successful observation data arrangement:
xoxox
oxoxo, which is just the negative of the "five point"
xoxox
symmetry. If other approximation schemes were used, it is quite possible that similar relationships between the approximation and the data point symmetries would be found.

Other than symmetry of data points, we must examine the two important straight lines near which the data points worked best. These are the y-axis and the axis of symmetry for the function U(x,y):



The y-axis is easiest to understand. The data near it contains the most significant digits and would therefore be best for fast convergence. The line of U symmetry is more difficult to understand. Here, if the data is piled right on the line, as shown on page 33, instead of a very rapid convergence we obtain convergence and then oscillation:



If we move the data points away from this axis slightly, then the convergence method works well (see page 33).

Thus, we have two major criteria for proper convergence defined and partially understood. For our sample problem LINEX we have determined the best location and the proper symmetrical arrangement of the observation data on U(x,y).

BIBLIOGRAPHY

1. Angel, Edward, "Inverse Boundary-Value Problems: Elliptic Equations," University of Southern California Engineering Publication, 1969.

2. Bellman, R. and R. Kalaba, Quasilinearization and Nonlinear Boundary-value Problems, American Elsevier, New York, 1965.

3. White, H.J. and S. Tauber, Systems Analysis, W.B. Saunders Company, Philadelphia, 1969.

4. Bellman, Richard, Adaptive Control Processes: A Guided Tour, Princeton University Press, 1961.

5. Lee, E. Stanley, Quasilinearization and Invarant Imbedding, Academic Press, New York and London, 1968.

6. Bellman, R.E. and S.E. Dreyfus, Applied Dynamic Programming, Princeton University Press, 1962.