



**National Library
of Canada**

**Bibliothèque nationale
du Canada**

Canadian Theses Service

Service des thèses canadiennes

**Ottawa, Canada
K1A 0N6**

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

Si manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

• UNIVERSITY OF ALBERTA

**A Local Mesh Refinement Technique
for
Multi-Grid Method on Elliptic Equations**

by



Allen K. Chien

A THESIS

**SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Master of Science**

IN

APPLIED MATHEMATICS

DEPARTMENT OF MATHEMATICS

EDMONTON, ALBERTA

FALL, 1989



**National Library
of Canada**

**Bibliothèque nationale
du Canada**

Canadian Theses Service Service des thèses canadiennes

**Ottawa, Canada
K1A 0N4**

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-64969-0

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: **Allen K. Chien**
TITLE OF THESIS: **A Local Mesh Refinement Technique**
 for Multi-Grid Method
 on Elliptic Equations
DEGREE FOR WHICH THESIS WAS PRESENTED: **Master of Science**
YEAR THIS DEGREE GRANTED: **1990**

PERMISSION IS HEREBY GRANTED TO THE UNIVERSITY OF ALBERTA LIBRARY TO REPRODUCE SINGLE COPIES OF THIS THESIS AND TO LEND OR SELL SUCH COPIES FOR PRIVATE, SCHOLARLY OR SCIENTIFIC RESEARCH PURPOSES ONLY.

THE AUTHOR RESERVES OTHER PUBLICATION RIGHTS, AND NEITHER THE THESIS NOR EXTENSIVE EXTRACTS FROM IT MAY BE PRINTED OR OTHERWISE REPRODUCED WITHOUT THE AUTHOR'S WRITTEN PERMISSION.



SIGNED

PERMANENT ADDRESS:

Department of Mathematics
University of Alberta
Edmonton, Alberta, Canada T6G 2G1

Date *August 3rd, 1990*

UNIVERSITY OF ALBERTA
FACULTY OF GRADUATE STUDIES AND RESEARCH

THE UNDERSIGNED CERTIFY THAT THEY HAVE READ, AND RECOMMEND TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH FOR ACCEPTANCE, A THESIS ENTITLED **A Local Mesh Refinement Technique for Multi-Grid Method on Elliptic Equations** SUBMITTED BY **Allen K. Chien** IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF **Master of Science** IN **APPLIED MATHEMATICS**.



Y.S. Wong
Supervisor



S. Sivaloganathan
Supervisor



B. Joe

Date June 18, 1990

To my grandmother

Abstract

In this thesis we investigate the performance of the multi-grid method in the context of local mesh refinements. The method illustrated here is based on the uniform multi-grid FAS scheme; however, the treatment of boundary conditions on internal boundaries introduced by local mesh refinement has been carefully considered. It allows us not only to obtain any desired refinement pattern but also to solve the discretized problems using a hierarchy of only uniform grids (extending over possibly smaller regions). Preliminary numerical experiments indicate that, apart from savings in storage, there are enormous savings in CPU time in comparison with standard multi-grid (where successive grids uniform refinements over the whole region of interest). The approach adopted here is especially useful for problems with boundary layers and in general for problems where rapid variation occurs in a very small region of the domain of interest while the behavior in the remaining part of the domain is smooth.

Acknowledgement

The author wishes to extend his appreciation to his supervisors, Dr. Y. S. Wong and Dr. S. Sivaloganathan, for all of their constant guidance, critical insight, and immeasurable contributions throughout this study. He also gratefully acknowledges the financial support for this research by the mentioned supervisors as well as the Department of Mathematics.

In addition, the author extends his gratitude to his fellow graduate students for having provided him a stimulating atmosphere to work in. He also would like to thank Mrs. Marion Benedict for her kindly help with TeX.

Finally, very special thanks are due his parents and sister for their understanding, encouragement, and support.

TABLE OF CONTENTS

	Page
1. Introduction	1
2. Uniform Multi-grid Method	4
2.1 Notations	4
2.2 Smoothing Property of Classical Iteration Methods	5
2.3 Uniform Multi-grid Algorithm	10
2.4 Nonlinear Treatment (FAS Scheme)	16
3. Local Mesh Refinement Technique	21
3.1 Introduction	21
3.2 Composite Grid Structure	23
3.3 Local Mesh Refinement Algorithm (LMRA)	25
3.4 Local Boundary Treatments	27
4. Numerical Results	33
Bibliography	37
Appendix: List of a FORTRAN77 Program for LMRA	40

1. Introduction.

Consider the elliptic differential equation:

$$\begin{cases} L_{\Omega} u = f_{\Omega}, & \text{in } \Omega \\ L_{\Gamma} u = f_{\Gamma}, & \text{on } \Gamma \end{cases} \quad (1.1)$$

where,

Ω is a d -dimensional region;

Γ is the boundary of Ω ;

$u = u(x)$, $f_{\Omega} = f_{\Omega}(x)$, $f_{\Gamma} = f_{\Gamma}(x)$ are defined on $x \in \Omega \cup \Gamma$; and

L_{Ω} , L_{Γ} are linear or nonlinear operators.

System (1.1) could be discretized somehow, say by the finite difference or finite element method, to get a discretized system:

$$L_h u_h = f_h, \quad x \in \Omega_h \subset \Omega \quad (1.2)$$

where $u_h, f_h \in G(\Omega_h)$: the space of gridfunctions defined on a grid Ω_h , and

$$L_h : G(\Omega_h) \rightarrow G(\Omega_h)$$

We assume here that the boundary conditions have been discretized and substituted into neighboring interior equations in the usual way.

As the model problem, let's consider the following two-dimensional Poisson equation with Dirichlet boundary conditions:

$$\begin{cases} \Delta u = f, & \text{in } \Omega = (0,1) \times (0,1) \\ u = g, & \text{on the boundary of } \Omega \end{cases} \quad (1.3)$$

Comparing (1.3) with (1.1), we have

$$L_{\Omega} = \Delta, \quad L_{\Gamma} = I, \quad \Omega = (0,1) \times (0,1)$$

Discretize the model problem (1.3) using a five-point difference stencil on the uniform grid Ω_h :

$$\Omega_h = \{(x_h, y_h) \mid (x_h, y_h) = (ih, jh), 0 \leq i, j \leq n\}$$

where the mesh size $h = \frac{1}{n}$, we have the corresponding discretized system (1.2) with

$$L_h = \frac{1}{h^2} \begin{pmatrix} A & I & & & \\ I & A & I & & \\ & \ddots & \ddots & \ddots & \\ & & I & A & I \\ & & & I & A \end{pmatrix}_{(n-1)^2 \times (n-1)^2} \quad (1.4)$$

and

$$A = \begin{pmatrix} -4 & 1 & & & \\ 1 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 1 \\ & & & 1 & -4 \end{pmatrix}_{(n-1) \times (n-1)} \quad (1.5)$$

Usually, due to the sparseness of the matrix L_h , iteration methods are preferable choices for solving the discretized system (1.2).

In this thesis we examine the multi-grid method, with a view to solving problems where the constraints of the computing facilities (costs and storage) render the use of a uniformly refined hierarchy of grids infeasible. The problems under consideration require a high resolution in very small regions of the domain, and in this context the most efficient approach may be to use a grid hierarchy where successive finer grids extend over possibly smaller regions of the preceding coarser grids. Brandt^{[4],[5]} and McCormick^[6] have used this approach. Their methods differ in the type of boundary conditions imposed on the artificial internal boundaries introduced by local mesh refinement. Our approach is a hybrid of these two approaches, i.e. we have adopted not only Brandt's intuitive implementation of the algorithm but also McCormick's careful internal boundary treatments. The reason for doing this is that we try to seek a

straight-forward implementation for the problem solving whilst at the same time trying to avoid introducing false physics into the solution.

Briefly, the contents of the thesis are as follows: in section 2, the smoothing property of classical iteration methods and uniform multi-grid method are discussed; then in section 3, the local mesh refinement technique is elaborated; next, some numerical results derived from using the technique are given in section 4; and finally, a program list (in *FORTRAN77*) for the local mesh refinement algorithm is attached in the Appendix.

2. Uniform Multi-grid Method.

For the time being, to facilitate the discussion, we assume that L_h in (1.2) is a linear operator, the nonlinear treatment will be presented in subsection 2.4.

2.1 Notations.

Let u_h^* be the exact solution of system (1.2), and v_h an approximation of u_h^* . There are two important measures of v_h as an approximation to u_h^* . The first one is the (algebraic) error e_h which is defined by:

$$e_h := u_h^* - v_h \quad (2.1)$$

Therefore, $e_h \in G(\Omega_h)$. Unfortunately, because u_h^* is unknown, the error e_h is just as inaccessible as the exact solution u_h^* itself. However, as an alternative measure, the residual given by:

$$r_h := f_h - L_h v_h \quad (2.2)$$

will give us a computable measure of how well v_h approximates u_h^* .

From (2.1), (2.2), and due to the uniqueness of the solution u_h^* , it is clear that the residual $r_h = 0$ if and only if the error $e_h = 0$. Nevertheless, it is not true that there is a direct connection between the error and residual, i.e. the error e_h may not be small when the residual r_h is small.

Now, by subtracting equation

$$L_h v_h = f_h - r_h$$

from

$$L_h u_h^* = f_h$$

we have the following residual equation:

$$L_A e_A = r_A \quad (2.3)$$

which provides us with the very important fact that the error e_A satisfies the same set of equations as the unknown u_A , when f_A is replaced by the residual r_A . It should be emphasized that the residual equation (2.3) plays a vital role in multi-grid methods which will be discussed shortly.

By solving the residual equation (2.3) for e_A , we may improve the approximation v_A by a new approximation $\bar{v}_A$ defined by:

$$\bar{v}_A := v_A + e_A \quad (2.4)$$

This is just the residual correction idea used by the classical iteration methods. Now, in order to understand the motivation of the multi-grid method, we examine the smoothing property of classical iteration methods.

2.2 Smoothing property of classical iteration methods.

Among the classical iteration methods, we focus on the damped Jacobi iteration method because it allows the most transparent analysis.

Let D_A denote the diagonal part of L_A , i.e.

$$D_A := \text{diag}(L_A)$$

The damped Jacobi iteration method for (1.2) is:

$$v_A^{m+1} := v_A^m + \omega D_A^{-1} (f_A - L_A v_A^m) \quad (2.5)$$

where, v_A^m is the m th approximation of v_A to u_A^1 on grid Ω_A , and $\omega \in [0, 1]$ is a damping parameter.

For exact solution of (1.2), we have:

$$u_h^* := u_h^* + \omega D_h^{-1}(f_h - L_h u_h^*) \quad (2.6)$$

By subtracting (2.5) from (2.6), we get:

$$e_h^{m+1} := (I - \omega D_h^{-1} L_h) e_h^m \quad (2.7)$$

where,

$$e_h^m := u_h^* - v_h^m$$

Denote the iteration matrix as S_h , i.e.

$$S_h := I - \omega D_h^{-1} L_h \quad (2.8)$$

we have:

$$\begin{aligned} e_h^{m+1} &= S_h e_h^m \\ &= S_h^2 e_h^{m-1} \\ &\dots \\ &= S_h^{m+1} e_h^0 \end{aligned} \quad (2.9)$$

Therefore,

$$\|e_h^m\| \leq \|S_h\|^m \cdot \|e_h^0\|$$

Practically speaking, the most commonly used vector (or matrix) norm $\|\cdot\|_\infty$ or $\|\cdot\|_2$ may be used here as measure for e_h (or S_h). It follows that if $\|S_h\| < 1$, then the error approaches zero as the iteration proceeds.

Recall the definition of spectral radius

$$\rho(S_h) := \max |\lambda_i(S_h)| \quad (2.10)$$

where $\lambda_i(S_A)$ is the i th eigenvalue of S_A , the following is true:

$$\lim_{m \rightarrow \infty} S_A^m = 0 \quad \text{if and only if} \quad \rho(S_A) < 1$$

Therefore, it follows that the error e_A^m converges to zero if and only if the spectral radius of S_A is less than one.

The term $\rho(S_A)$ is also called the convergence factor and could be interpreted as the worst factor by which the error is reduced by each relaxation iteration. Moreover, suppose that $\rho(S_A) < 1$ and let M be the smallest integer satisfying

$$[\rho(S_A)]^M \leq 10^{-t}$$

by solving for M , we have:

$$M \geq \frac{t}{-\log_{10}[\rho(S_A)]} \quad (2.11)$$

Therefore, the convergence factor $\rho(S_A)$ could be used to calculate the number of iterations needed to reduce the error by a factor of 10^{-t} . The term $-\log_{10} [\rho(S_A)]$ on the denominator in (2.11) is called the convergence rate which gives the number of iterations required to reduce the error by one decimal digit. It is obvious from (2.11) that the smaller the value of $\rho(S_A)$, the higher the convergence rate of iterations.

Now, let's take a look at the convergence behavior of our model problem in more detail.

The eigenvalues and eigenfunctions of L_A defined by (1.4) are

$$\lambda_{pq}(L_A) = -\frac{4}{h^2} \left[\sin^2\left(\frac{p\pi h}{2}\right) + \sin^2\left(\frac{q\pi h}{2}\right) \right] \quad (2.12)$$

$$w_{pq}(L_A) = \sin(p\pi x_A) \sin(q\pi y_A), \quad (x_A, y_A) \in \Omega_A \quad (2.13)$$

for $1 \leq p, q \leq n-1$.

By (2.8), we have that S_h has the same eigenfunctions as L_h , i.e.

$$w_{pq}(S_h) = w_{pq}(L_h) = \sin(p\pi x_h) \sin(q\pi y_h), \quad (x_h, y_h) \in \Omega_h$$

and the corresponding eigenvalues of S_h are:

$$\lambda_{pq}(S_h) = 1 - \omega \left[\sin^2\left(\frac{p\pi h}{2}\right) + \sin^2\left(\frac{q\pi h}{2}\right) \right] \quad (2.14)$$

for $1 \leq p, q \leq n-1$.

It is clear that for convergence of damped Jacobi iteration, it is necessary to have $0 < \omega \leq 1$.

By using the eigenfunction expansion for e_h^0 along $\{w_{pq}(L_h)\}$, we have:

$$e_h^0 := \sum_{1 \leq p, q \leq n-1} \alpha_{pq} w_{pq}(L_h) \quad (2.15)$$

where α_{pq} are the proper weighting factors.

On the right hand side of (2.15), the components $\alpha_{pq} w_{pq}(L_h)$ corresponding to $1 \leq p, q \leq \frac{n}{2} - 1$ are called low frequency or smooth error components of e_h^0 while those corresponding to $\frac{n}{2} \leq p, q \leq n-1$ are called high frequency or oscillatory error components.^{[6],[7]}

Note that

$$w_{pq}(S_h) = w_{pq}(L_h)$$

and

$$S_h w_{pq}(S_h) = \lambda_{pq}(S_h) w_{pq}(S_h)$$

by (2.9), we have:

$$\begin{aligned} e_h^m &= S_h^m e_h^0 \\ &= S_h^m \sum_{1 \leq p, q \leq n-1} \alpha_{pq} w_{pq}(L_h) \end{aligned}$$

$$\begin{aligned}
&= \sum_{1 \leq p, q \leq n-1} \alpha_{pq} S_h^m w_{pq}(S_h) \\
&= \sum_{1 \leq p, q \leq n-1} \alpha_{pq} \lambda_{pq}^m(S_h) w_{pq}(S_h) \\
&= \sum_{1 \leq p, q \leq n-1} \alpha_{pq} \lambda_{pq}^m(S_h) w_{pq}(L_h) \\
&= \sum_{1 \leq p, q \leq n-1} \lambda_{pq}^m(S_h) [\alpha_{pq} w_{pq}(L_h)] \tag{2.16}
\end{aligned}$$

From (2.16), we may discover some very interesting facts. If we take $p = q = 1$ in (2.14), we have

$$\begin{aligned}
\lambda_{11}(S_h) &= 1 - \omega \left[\sin^2\left(\frac{\pi h}{2}\right) + \sin^2\left(\frac{\pi h}{2}\right) \right] \\
&= 1 - 2\omega \sin^2\left(\frac{\pi h}{2}\right) \\
&\approx 1 - 2\omega \left(\frac{\pi h}{2}\right)^2 \\
&= 1 - \frac{\omega \pi^2}{2} h^2 \\
&= 1 - O(h^2)
\end{aligned}$$

which approaches one when $h \rightarrow 0$ for any $0 < \omega \leq 1$. Therefore, by (2.16), we know that the error component $\alpha_{11}w_{11}(L_h)$ has remained almost undamped after several iterations for any $0 < \omega \leq 1$. In other words, no matter how we choose the damping parameter ω , the smooth error components can not be damped effectively. The smaller the mesh size h , the worse the convergence rate for smooth error components.

On the other hand, let's consider the smoothing effects of the oscillatory error components, say, take the extreme case $\lambda_{n-1, n-1}(S_h)$:

$$\begin{aligned}
\lambda_{n-1, n-1}(S_h) &= 1 - \omega \left[\sin^2\left(\frac{(n-1)\pi h}{2}\right) + \sin^2\left(\frac{(n-1)\pi h}{2}\right) \right] \\
&= 1 - 2\omega \sin^2\left(\frac{(n-1)\pi h}{2}\right)
\end{aligned}$$

If we choose $\omega = \frac{1}{2}$, then

$$\begin{aligned}\lambda_{n-1,n-1}(S_h) &= 1 - \sin^2\left(\frac{(n-1)\pi h}{2}\right) \\ &= \cos^2\left(\frac{(n-1)\pi h}{2}\right) \\ &= \sin^2\left(\frac{\pi h}{2}\right) \\ &= O(h^2)\end{aligned}$$

Therefore, by choosing an appropriate ω , the damped Jacobi iteration method is very efficient in reducing high frequency error components.

Hence, we observe that the damped Jacobi iteration method works very well for the first several iterations to damp the oscillatory error components; however, once the oscillatory errors have been removed, the iteration is much less effective in reducing the remaining smooth error components. We call this property the 'smoothing property' of the damped Jacobi iteration method.

Although we have derived the above smoothing property only for the damped Jacobi iteration method, most of the classical iteration methods have this property, too.^[21]

Hence, the motivation is to search for an 'optimal' method with the following properties: first, it should damp the oscillatory error components as effectively as the classical iteration methods; second, with some auxiliary treatment, it should eliminate the smooth error components almost as effectively as the oscillatory ones; and finally, for optimality, we want to solve the problem in $O(m)$ work units (where m is the size of the problem), the convergence rate should be independent of mesh size h (or at least asymptotically). We will see that multi-grid methods meet the above criteria.

2.3 Uniform multi-grid algorithm.

The previous subsection showed that most of the classical iteration methods such as damped Jacobi method (with $\omega = \frac{1}{2}$) are quite efficient in smoothing out the oscillatory error components. However, they are much less effective in reducing the smooth error components. This is true especially for the case when the grid size h approaches to zero.

Therefore, in order to damp low frequency errors, we should construct a complementary iteration which can reduce the smooth error components very well. By combining this complementary iteration with certain classical iteration method, we may expect to have powerful algorithm to reduce both the oscillatory and smooth error components effectively.

Such a complementary iteration can be obtained by means of the coarse grid $\Omega_H \subset \Omega_h \subset \Omega$. Generally speaking, we may choose the grid size H such that $H = 2h$, where h is the grid size of its immediate finer grid Ω_h . The reason that we construct such complementary iteration is as follows.

Let $S_h : G(\Omega_h) \rightarrow G(\Omega_h)$ be some classical iteration operator, and v_h be an approximation of u_h^* . Then,

$$\bar{v}_h := S_h(v_h, f_h) \quad (2.17)$$

Denote

$$e_h := u_h^* - \bar{v}_h$$

we have

$$L_h e_h = r_h \quad (2.18)$$

where,

$$r_h := f_h - L_h \bar{v}_h \quad (2.19)$$

Hence, if we can solve (2.18) exactly to get e_A^* , we have achieved our aim because

$$u_A^* = v_A + e_A^*$$

Unfortunately, to solve (2.18) exactly is as difficult as to solve (1.2). Nevertheless, by the previous discussion, we know that e_A is a smooth function, and so is r_A . Therefore, both of them can be well represented on the coarser grid. It follows that the transferring of the residual to the coarser grid can be done without much loss of the information. Hence, the smooth error components on the fine grid now appear more oscillatory on a coarser grid, and it can be removed more efficiently by classical iterations.

Thus, we have the following *coarse-grid correction algorithm*:

Given approximation v_A to u_A^* ,

(1) Calculation of residual:

$$r_A := f_A - L_A v_A;$$

(2) Restriction of residual:

$$r_H := R_A^H r_A;$$

(3) Exact solution of coarse-grid equation:

$$e_H := L_H^{-1} r_H;$$

(4) Prolongation of correction:

$$e_A := I_H^A e_H;$$

(5) Correction of v_A :

$$v_A := v_A + e_A$$

where,

$R_A^H : G(\Omega_A) \rightarrow G(\Omega_H)$ is the restriction operator;

$I_H^A : G(\Omega_H) \rightarrow G(\Omega_A)$ is the interpolation operator; and

L_H is the discretised operator of L on grid Ω_H .

Unfortunately, the coarse-grid correction by itself is not convergent since the spectral radius of the coarse grid correction iteration matrix is greater than or equal to 1. However, the two grid method, which is the combination of certain classical iteration method with the coarse-grid method, does give a rapid convergence rate. Moreover, the convergence rate is independent of grid size h (or at least asymptotically).^[11]

The algorithm of the two-grid method is as follows:

Given approximation v_h to u_h^* ,

(1) Pre-relaxation (ν_1 times):

$$\bar{v}_h := S_h^{\nu_1}(v_h, f_h);$$

(2) Coarse-grid correction:

$$(2.a) \quad r_h := f_h - L_h \bar{v}_h;$$

$$(2.b) \quad r_H := R_h^H r_h;$$

$$(2.c) \quad e_H := L_H^{-1} r_H;$$

$$(2.d) \quad e_h := I_h^H e_H;$$

$$(2.e) \quad \bar{v}_h := \bar{v}_h + e_h.$$

Due to the fact that step (2.d) may introduce some oscillatory error components back to e_h , we may add another ν_2 relaxations (called post-relaxation) to the algorithm, i.e.

(3) Post-relaxation (ν_2 times):

$$\bar{v}_h := S_h^{\nu_2}(\bar{v}_h, f_h).$$

Now, let's look at the two-grid algorithm stated above in more detail. First of all, for steps (1) and (3), due to the smoothing property of most

classical iterations, there is no significant advantage to choose certain method over another among the smoothing methods. Empirically speaking, except for some complicated cases, pointwise or linewise symmetric Gauss-Seidel iterations are the usual choices. Moreover, different iterations may be used for steps (1) and (3), respectively. However, for ease of program implementation, the same smoothing method is frequently used for both steps.

Next, for the restriction operator R_h^H , there are several natural choices. The simplest one is called injection. It is just the case that v_H on the coarse grid simply takes its value directly from the corresponding fine grid point v_h . Therefore, for one-dimensional problems,

$$v_H(iH) := v_h(2ih), \quad 0 \leq i \leq \frac{N}{2}$$

and for two-dimensional problems, we have:

$$v_H(iH, jH) := v_h(2ih, 2jh), \quad 0 \leq i, j \leq \frac{N}{2} \quad (2.20)$$

An alternative choice is called full-weighting. That is, for one-dimensional problems, it is defined as:

$$v_H(jH) = \frac{1}{4}(v_h((2j-1)h) + 2v_h(2jh) + v_h((2j+1)h)), \quad 1 \leq j \leq \frac{N}{2} - 1$$

and for two-dimensional ones, the corresponding relation is:

$$\begin{aligned} v_H(iH, jH) := & \frac{1}{16}(v_h((2i-1)h, (2j-1)h) + v_h((2i-1)h, (2j+1)h) \\ & + v_h((2i+1)h, (2j-1)h) + v_h((2i+1)h, (2j+1)h) \\ & + 2(v_h(2ih, (2j-1)h) + v_h(2ih, (2j+1)h) \\ & + v_h((2i-1)h, 2jh) + v_h((2i+1)h, 2jh)) \\ & + 4v_h(2ih, 2jh)), \quad 1 \leq i, j \leq \frac{N}{2} - 1 \end{aligned} \quad (2.21)$$

It should be pointed out that unless the coefficients of the problem vary wildly, injection is the natural and convenient choice for restriction.

Thirdly, for prolongation operator I_H^h in step (2.d), in order to have the full efficiency, the order of I_H^h should be no less than the order of the differential equation. Nevertheless, practically speaking, higher prolongation order has no obvious advantage. Therefore, for second order problems, corresponding to one- or two-dimensional problems, the linear or bi-linear interpolation is the common choice, i.e. for one-dimensional problems, the prolongation is:

$$\begin{cases} v_h(2jh) := v_H(jH), & 0 \leq j \leq \frac{N}{2} \\ v_h((2j+1)h) := \frac{1}{2}(v_H(jH) + v_H((j+1)H)), & 0 \leq j \leq \frac{N}{2} - 1 \end{cases}$$

and for two-dimensional problems, we have

$$\begin{cases} v_h(2ih, 2jh) := v_H(iH, jH), & 0 \leq i, j \leq \frac{N}{2} \\ v_h((2i+1)h, 2jh) := \frac{1}{2}(v_H(iH, jH) + v_H((i+1)H, jH)), \\ v_h(2ih, (2j+1)h) := \frac{1}{2}(v_H(iH, jH) + v_H(iH, (j+1)H)), \\ v_h((2i+1)h, (2j+1)h) := \frac{1}{4}(v_H(iH, jH) + v_H((i+1)H, jH) \\ \quad + v_H(iH, (j+1)H) + v_H((i+1)H, (j+1)H)), \\ 0 \leq i, j \leq \frac{N}{2} - 1 \end{cases} \quad (2.22)$$

In step (2.c), the operator L_H is the Ω_H approximation to L_Ω and L_Γ . It could be obtained by a similar discretization as L_h on the coarser grid Ω_H . Furthermore, by looking at step (2.c) more carefully, we can immediately discover that, although it needs much less work than doing that on the finer level Ω_h , it is still very expensive to solve the equation $L_H e_H = r_H$ exactly if the grid Ω_H is not coarse enough. However, the idea of the two-grid method provides us with a very natural way to overcome the difficulty here. We may solve $L_H e_H = r_H$ by using the two-grid method on an even coarser grid Ω_{2H} . This

idea can be carried out recursively until we reach the coarsest grid Ω_{N_0} . Then, on that coarsest grid, it should be quite cheap to solve for e_{N_0} exactly.

Finally, there is one important remaining problem to be solved, that is how to get a "good" initial approximation v_h on the grid Ω_h . A good initial approximation should, at least in its early stages, improve the relaxation scheme. A well-known technique to obtain such a good initial approximation is to apply some preliminary iteration on a coarser grid and then interpolate the resulting approximation to the original fine grid. Relaxation on a coarser grid is less expensive since there are fewer unknowns to be updated. Moreover, since the convergence factor behaves like $1 - O(H^2)$ on the coarser grid Ω_H , the coarser grid will have a fairly improved convergence rate.

We have mentioned before that the convergence rate of the two-grid method is independent of mesh spacing. Now, because a multi-grid method is one in which the two-grid method is applied recursively over a sequence of grids, we may expect that it attains grid independent convergence rates, too; and indeed, that is the case.^[11]

Figure 2.1 summarizes in flow diagram form the uniform multi-grid algorithm for linear problems.

2.4 Nonlinear treatment and FAS algorithm.

The previously described multi-grid method is only valid for the case when L is linear. However, many of the problems we wish to solve, for example the Navier-Stokes equations, are nonlinear. Therefore, we have to extend the multi-grid approach to treat nonlinear problems.

One approach to deal with nonlinear problems is to use a Newton multi-grid iteration method.^{[12],[13],[14]} Nevertheless, instead of solving nonlinear problems directly, we have to spend considerable effort in carrying out the Newton

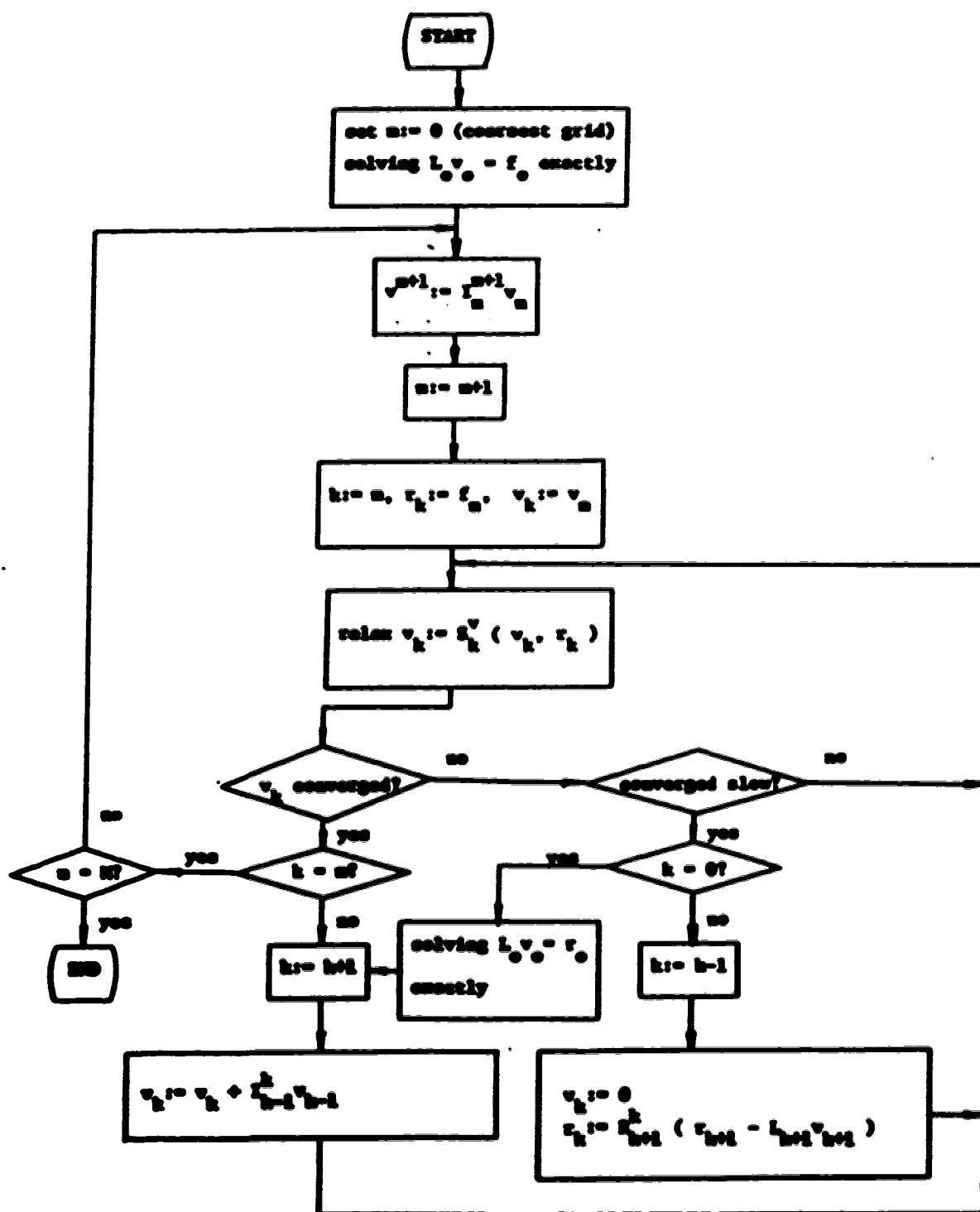


Figure 2.1

Uniform multi-grid algorithm for linear problems

linearization. The direct approach is to use the Full Approximation Scheme (or FAS). It was outlined by Brandt.^{[6],[7]} For our later use, we present the scheme here.

For convenience, we rewrite (1.2) here:

$$L_h u_h = f_h, \quad x \in \Omega_h \subset \Omega \quad (2.23)$$

where, $u_h, f_h \in G(\Omega_h)$, and

$$L_h : G(\Omega_h) \rightarrow G(\Omega_h)$$

is a nonlinear operator.

The residual equation is as follows:

$$\tilde{L}_h e_h = r_h \quad (2.24)$$

where,

$$\tilde{L}_h e_h \equiv L_h(v_h + e_h) - L_h v_h \quad (2.25)$$

and

$$r_h \equiv f_h - L_h v_h.$$

By (2.25), we know that $\tilde{L}_h = L_h$ if L_h is linear.

From the previous discussion, we know that the error e_h and the residual r_h are smooth functions. Therefore, similar to the linear situation before, we would like to approximate them on the coarser grid Ω_H , where $\Omega_H \subset \Omega_h \subset \Omega$. The approximation to (2.24) on Ω_H is:

$$L_H(R_h^H v_h + e_H) - L_H(R_h^H v_h) = R_h^H r_h \quad (2.26)$$

where L_H is the approximation to L on the coarse grid Ω_H .

Let

$$v_H := R_h^H v_h + e_H$$

$$f_h^H := L_H(R_h^H v_h) + R_h^H \tau_h$$

by (2.26), we have

$$L_H v_H = f_h^H \quad (2.27)$$

Therefore, rather than solve the error e_H , we compute v_H , the full solution on Ω_H . Once we get the approximation v_H , we prolong the error and update v_h

$$v_h := v_h + I_h^h(v_H - R_h^H v_h) \quad (2.28)$$

which we may expect to be a better approximation v_h to u_h^* .

Now, let's go back to look at (2.27) more carefully. In fact, the right hand side of (2.27) f_h^H can be expressed as

$$f_h^H := f_H + \tau_h^H \quad (2.29)$$

where,

$$\tau_h^H := L_H(R_h^H v_h) - R_h^H(L_h v_h) \quad (2.30)$$

By (2.30), we know that term τ_h^H actually is the truncation error on Ω_H relative to Ω_h . Furthermore, if we consider (2.29) as such a treatment that brings more residual information on fine grid Ω_h down to the coarser grid Ω_H , we may expect to have an approximation on Ω_H with the accuracy of the fine grid Ω_h . It is very important to mention that this idea is the key point of our local mesh refinement techniques.

Figure 2.2 illustrates the FAS scheme coupled with the full multi-grid approach (FMG).

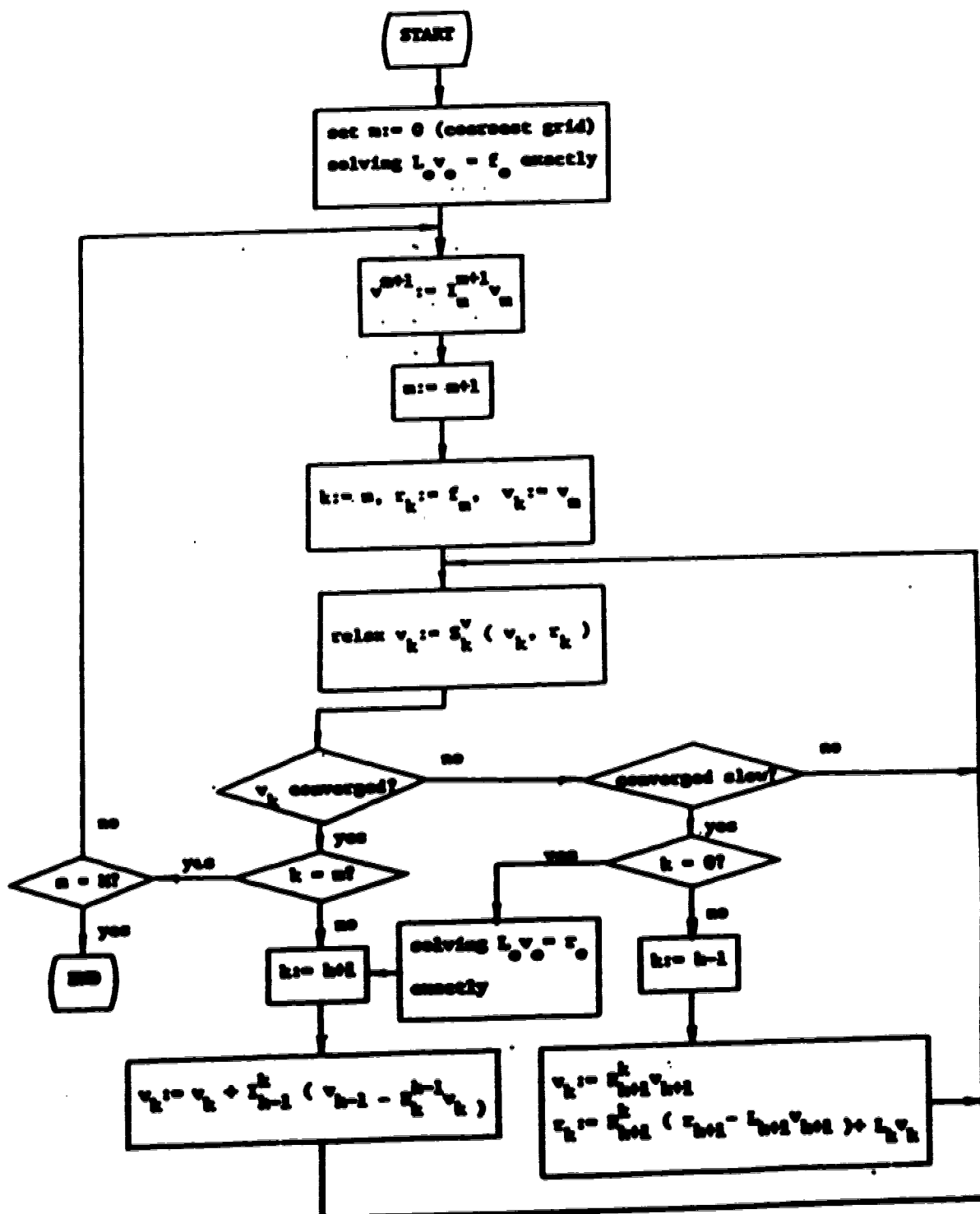


Figure 2.2
Uniform multi-grid FAS algorithm

3. Local Mesh Refinement Technique.

3.1 Introduction.

For many problems, because of the properties of the problems themselves (say, with the unbounded domain, containing singular points such as sources or sinks, or with discontinuous solutions or shock phenomena), or because of the constraints of our computing facilities (memory, project funds, etc.), different resolution in different parts of the domain should be considered.

For example, let's consider our model problem (1.3)

$$\begin{cases} \Delta u = f, & \text{in } \Omega = (0,1) \times (0,1) \\ u = g, & \text{on the boundary of } \Omega \end{cases} \quad (3.1)$$

with

$$f = [4\alpha^2((1-x)^2 + (1-y)^2) - 4\alpha] \cdot \exp[-\alpha((1-x)^2 + (1-y)^2)]$$

and

$$g = \begin{cases} \exp[-\alpha(1 + (1-y)^2)], & \text{if } x=0 \text{ and } 0 \leq y \leq 1; \\ \exp[-\alpha(1-y)^2], & \text{if } x=1 \text{ and } 0 \leq y \leq 1; \\ \exp[-\alpha(1 + (1-x)^2)], & \text{if } 0 \leq x \leq 1 \text{ and } y=0; \\ \exp[-\alpha(1-x)^2], & \text{if } 0 \leq x \leq 1 \text{ and } y=1. \end{cases}$$

where α is a parameter.

We know that the analytical solution of (3.1) is:

$$u(x,y) = \exp[-\alpha((1-x)^2 + (1-y)^2)] \quad (3.2)$$

However, when α is large, say $\alpha = 100$, the solution $u(x,y)$ will change dramatically near the point $(1,1)$ while it will remain pretty smooth elsewhere (almost zero). Then, it is very important to look closely at the solutions near point $(1,1)$. Thus, the region near $(1,1)$ should be refined.

When we consider areas for local mesh refinement, there are several factors that should be taken into consideration.^[4] First, introducing such a local mesh

refinement should not increase the computational cost dramatically. Second, it should not entail reconsideration of the problem entirely when we introduce the local mesh refinement. Next, introducing the local mesh refinement should allow for effective self-adaptive treatments. Finally, we should avoid introducing any false physics into the solutions when we use the local mesh refinement technique.

How about choosing an arbitrary general grid? Practically speaking, although it might be quite flexible, it is very complicated to discretize continuous problems on such a grid, let alone to solve them. Moreover, due to the complicated and tedious bookkeeping work for the grid information, a large amount of memory has to be supplied and much CPU time has to be spent on the extra data management work. The finite element method is a particular example of such a grid structure. Thus, it doesn't seem worthwhile to incorporate such a general grid structure to handle local mesh refinement.

In fact, it is not necessary to use a general grid. The following local mesh refinement technique is flexible enough to permit any desired refinement pattern and to obtain the solution of the discretized problem by applying the multi-grid method on uniform grids only.

Recall that for the uniform multi-grid method, the convergence rate is independent of grid size h and the accuracy is improved with grid refinement. Therefore, a natural idea arises: in order to get an accurate picture of the global solution structure, we refine some regions locally to obtain higher accuracy in those regions. Is there a possible modification to the uniform multi-grid method so that it could meet the following requirements:

- (1) to achieve the desired accuracy throughout the entire domain;
- (2) to solve the problems on uniform grids only; and

(3) to attain the mesh independent convergence rate (even though the different mesh sizes may appear in the different regions of domain)?

The method introduced here uses the multi-grid method on the so-called "composite-grid", i.e. the union of a series of uniform grids, and is basically similar to Brandt's MLAT algorithm;^[4,7] however, in order to avoid introducing any false physics into the solutions, the flux balance method has been introduced to treat the residuals on the local boundaries. As we will see, by using such local boundary treatment, the whole nonuniform multi-grid iteration will be conservative. This enhances the convergence of the multi-grid method, and provides a more efficient approach than standard multi-grid.

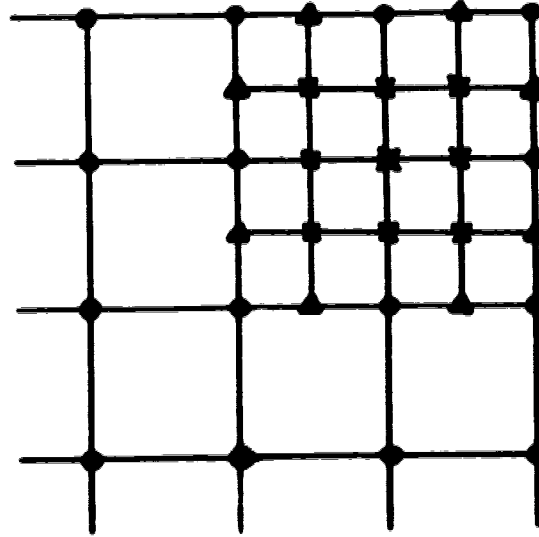
3.2 Composite grid structure.

Now, let's look at the structure of our composite grid. For simplicity, we assume that in the domain of our problem, there is only one local region that should be refined. We organize the composite grid as the combination of a series of uniform grids, say, $\Omega_0, \Omega_1, \dots, \Omega_M$, where Ω_0 is the coarsest grid and Ω_M is the finest grid. As usual, we choose grid size $h_k = 2h_{k+1}$; moreover, let every other grid line of Ω_{k+1} be the grid line of Ω_k . However, unlike the situation in the uniform multigrid method, grid Ω_{k+1} here may only cover part of Ω_k region, i.e. the region that needs to be refined.

Taking our model problem (3.1) as an example, we may construct our composite grid near (1,1) as shown in Figure 3.1 (for clear illustration, only two grids Ω_k and Ω_{k+1} have been shown here). We have:

$$\Omega_{\text{com}} = \Omega_0 \cup \Omega_1 \cup \dots \cup \Omega_k \cup \Omega_{k+1} \cup \dots \cup \Omega_M \quad (3.3)$$

where, Ω_0 is the coarsest grid and Ω_M is the finest grid.



$$\begin{cases} \Omega_k = oU \circ U \# \\ \Omega_{k+1} = oU \Delta U \# U \times \end{cases}$$

Figure 3.1

Composite grid structure

It is very important to emphasize the role that Ω_k plays. In the regions that the local mesh refinement has to be considered, i.e. if there is a finer grid Ω_{k+1} , Ω_k can be treated as a coarse grid for the correction purpose. On the other hand, in those regions where there is no further resolution needed, Ω_k can be treated as the finest grid there. For example, in *figure 3.1*, those points with $\circ$ or $\#$ on grid Ω_k can be considered as the coarse grid points of grid Ω_{k+1} , and those points marked with $\circ$ can be treated as the finest grid points on grid Ω_k . The first point of view here is especially useful in our local mesh refinement technique.

Note here that in the composite grid Ω_{com} , although Ω_k and Ω_{k+1} share the common geometric points $\circ$ and $\#$; generally speaking, the values at those points on different levels Ω_k and Ω_{k+1} are different. On the other hand, just because of those points, we are able to transfer the residuals from local region

grid Ω_{k+1} to coarser grid Ω_k ; and also because of those points, we are able to bring the correction back to the local region Ω_{k+1} to achieve higher accuracy. Therefore, they are the key links within the composite grid Ω_{com} . Figure 3.2 gives us an intuitive picture of this point.

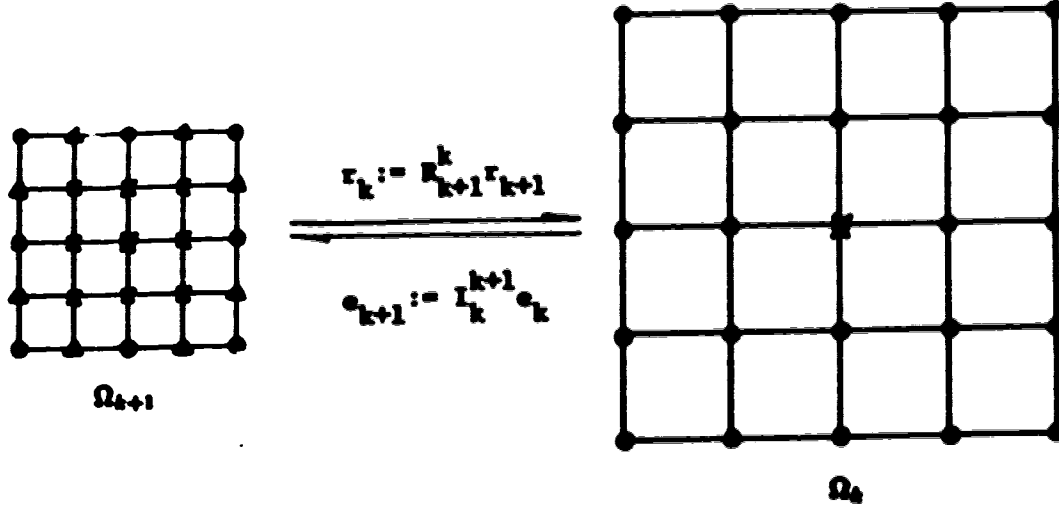


Figure 3.2

3.3 Local mesh refinement algorithm (LMRA).

Recall that the uniform multi-grid FAS algorithm can be summarized as follows:

For a given discretized system

$$L_{k+1}v_{k+1} = f_{k+1}, \quad x \in \Omega_{k+1} \subset \Omega \quad (3.4)$$

and an approximation v_{k+1} to u_{k+1}^* ,

(1) Pre-relaxation (ν_1 times):

$$v_{k+1} := S_{k+1}^{\nu_1}(v_{k+1}, f_{k+1}) \quad (3.5)$$

(2) Coarse-grid correction:

$$v_k := L_k^{-1}(L_k(R_{k+1}^k v_{k+1}) + R_{k+1}^k(f_{k+1} - L_{k+1}v_{k+1})) \quad (3.6)$$

(3) Correction:

$$v_{h+1} := v_{h+1} + I_h^{h+1}(v_h - R_{h+1}^h v_{h+1}) \quad (3.7)$$

(4) Post-relaxation (ν_2 times):

$$v_{h+1} := S_{h+1}^{\nu_2}(v_{h+1}, f_{h+1}) \quad (3.8)$$

Notice that the right hand side of (3.7) can be express as: $L_h^{-1}(\tau_h^{h+1} + R_{h+1}^h f_{h+1})$, where

$$\tau_h^{h+1} := L_h(R_{h+1}^h v_{h+1}) - R_{h+1}^h(L_{h+1} v_{h+1}) \quad (3.9)$$

It is clear that the term τ_h^{h+1} is the truncation error on Ω_h relative to Ω_{h+1} . Thus, with the help of the composite grid, the idea above results in the following *local mesh refinement algorithm (LMRA)* on the composite grid Ω_{com} :

Given an approximation v_{h+1} to u_{h+1}^* on the local region grid Ω_{h+1} ,

(1) Pre-relaxation on local region grid Ω_{h+1} (ν_1 times):

$$v_{h+1} := S_{h+1}^{\nu_1}(v_{h+1}, f_{h+1});$$

(2) Coarse-grid correction:

(a) Calculate residuals on the inner points of local region Ω_{h+1}

(corresponding to points x and u):

$$r_{h+1} := f_{h+1} - L_{h+1} v_{h+1};$$

(b) By using the flux balance method, calculate residuals on those local boundary points corresponding to c :

$$r_{h+1} := f_{h+1} - \bar{L}_{h+1}(v_{h+1}, v_h)$$

where $\bar{L}_{h+1} : G(\Omega_{com}) \rightarrow G(\Omega_{com})$ is an approximation to L on Ω_{com} ;

(c) Restrict all residuals obtained by both (a) and (b) to coarse grid Ω_h :

$$r_h := R_{h+1}^h r_{h+1};$$

(d) Solve v_h on coarse grid Ω_h :

$$v_h := L_h^{-1}(L_h(R_{h+1}^h v_{h+1}) + r_h);$$

(3) Correction:

$$v_{h+1} := v_{h+1} + I_h^{h+1}(v_h - R_{h+1}^h v_{h+1});$$

(4) Post-relaxation (ν_2 times):

$$v_{h+1} := S_{h+1}^{\nu_2}(v_{h+1}, f_{h+1}).$$

There are some comments about the algorithm above. First of all, notice that in each step, we solve our problems on uniform grids only, that means that we discretize the continuous problems and make the relaxation sweeps on the uniform grids only. It definitely makes the problem solving much easier than the use of any general grid structures.

Secondly, the algorithm does fit self-adaptive needs. We may detect those local regions which need to have higher resolutions by applying Richardson extrapolation on the error^[6] or by some optimization technique on the mesh size and problem approximation order.^{[5],[9],[7]}

Next, the first approximation v_{h+1} to u_{h+1}^* on Ω_{h+1} could be obtained by the linear or bi-linear interpolation from solution v_h on grid Ω_h . As we did before for the uniform multi-grid algorithm, we may get v_h by using the full multi-grid method.

Finally, in the coarse-grid correction step, the calculation of the residuals on local boundaries is the key role of our local mesh refinement technique. Thus, we are going to look at it very carefully in the next subsection.

3.4 Local boundary treatments.

For the residuals on the inner points of local mesh regions, we may use the same difference star for L_h as we do on the coarse grid Ω_h . Let's take our

model problem as an example. We use the finite volume flux balance method to get the five-point finite difference discretization of Δu on the inner points of local mesh region. For any finite volume $S \subset \Omega$, let ∂S be the boundary of S . Then, by Green's Theorem, we have

$$\iint_S \Delta u \, dx \, dy = \int_{\partial S} \frac{\partial u}{\partial n} \, dl \quad (3.10)$$

where n is the normal direction of the boundary ∂S .

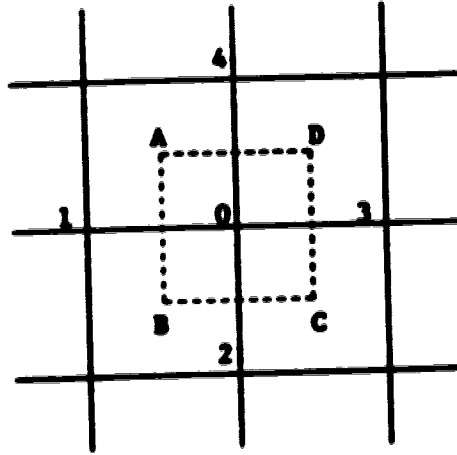


Figure 3.3

Finite volume method for inner points

By figure 3.3, for inner points x and n of the local region, we choose the finite volume S_0 as the area enclosed by ∂S_0 which is $ABCD$. Now, by using the following discretization to approximate $\int_{\partial S_0} \frac{\partial u}{\partial n} \, dl$:

$$\begin{aligned} \int_{\partial S_0} \frac{\partial u}{\partial n} \, dl &= \int_{AB} \frac{\partial u}{\partial n} \, dl + \int_{BC} \frac{\partial u}{\partial n} \, dl + \int_{CD} \frac{\partial u}{\partial n} \, dl + \int_{DA} \frac{\partial u}{\partial n} \, dl \\ &\approx \frac{u_{i+1}^{(1)} - u_{i+1}^{(0)}}{h_{i+1}} \cdot h_{i+1} + \frac{u_{i+1}^{(2)} - u_{i+1}^{(0)}}{h_{i+1}} \cdot h_{i+1} \\ &\quad + \frac{u_{i+1}^{(3)} - u_{i+1}^{(0)}}{h_{i+1}} \cdot h_{i+1} + \frac{u_{i+1}^{(4)} - u_{i+1}^{(0)}}{h_{i+1}} \cdot h_{i+1} \\ &= u_{i+1}^{(1)} + u_{i+1}^{(2)} + u_{i+1}^{(3)} + u_{i+1}^{(4)} - 4u_{i+1}^{(0)} \end{aligned} \quad (3.11)$$

and from (3.1) and (3.10), we have the following five-point finite difference equation:

$$v_{k+1}^{(1)} + v_{k+1}^{(2)} + v_{k+1}^{(3)} + v_{k+1}^{(4)} - 4v_{k+1}^{(0)} = f_{k+1}^{(0)} \quad (3.12)$$

where,

$$f_{k+1}^{(0)} = \iint_{S_0} f \, dx \, dy.$$

(The superscripts above correspond to the number of geographic positions.)

However, for the local boundary points of the composite grid (but not on the corners), because of the different grid sizes presented in the composite grid Ω_{com} , we can not use the same strategy above to treat the residuals here. Therefore, a special treatment should be applied. Furthermore, in order to avoid introducing any false physics, we have to force the discretized system to be conservative. Thus, we construct the finite volume as illustrated in Figure 3.4.

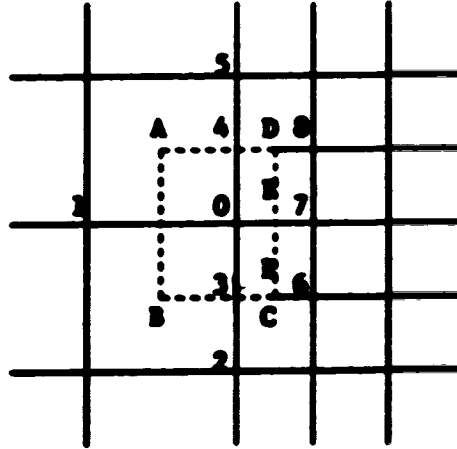


Figure 3.4

**Seven-point finite volume method
for inner boundary (but not corner) points**

From Figure 3.4, we have:

$$\int_{\partial S_0} \frac{\partial u}{\partial n} d\ell = \int_{AB} \frac{\partial u}{\partial n} d\ell + \int_{BC} \frac{\partial u}{\partial n} d\ell + \int_{CD} \frac{\partial u}{\partial n} d\ell + \int_{DA} \frac{\partial u}{\partial n} d\ell$$

Then, we approximate $\int_{AB} \frac{\partial u}{\partial n} d\ell$ by $\frac{v_b^{(1)} - v_{b+1}^{(0)}}{h_b} \cdot h_b$, $\int_{BC} \frac{\partial u}{\partial n} d\ell$ by $\frac{v_{b+1}^{(2)} - v_{b+1}^{(0)}}{2h_{b+1}} \cdot (\frac{h_b}{2} + \frac{h_{b+1}}{2})$, and $\int_{DA} \frac{\partial u}{\partial n} d\ell$ by $\frac{v_{b+1}^{(5)} - v_{b+1}^{(0)}}{2h_{b+1}} \cdot (\frac{h_b}{2} + \frac{h_{b+1}}{2})$. For $\int_{CD} \frac{\partial u}{\partial n} d\ell$, however, we can not simply approximate it by $\frac{v_{b+1}^{(7)} - v_{b+1}^{(0)}}{h_{b+1}} \cdot h_{b+1}$ because it will destroy the conservation of the discrete system.^[6] Therefore, we modify it by letting

$$\int_{CD} \frac{\partial u}{\partial n} d\ell \approx \frac{v_{b+1}^{(6)} - v_{b+1}^{(2)}}{h_{b+1}} \cdot \frac{1}{2} h_{b+1} + \frac{v_{b+1}^{(7)} - v_{b+1}^{(0)}}{h_{b+1}} \cdot h_{b+1} + \frac{v_{b+1}^{(8)} - v_{b+1}^{(4)}}{h_{b+1}} \cdot \frac{1}{2} h_{b+1} \quad (3.13)$$

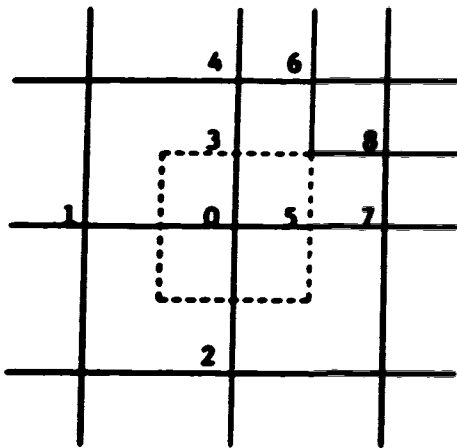
This yields the following approximation $L_{b+1}(v_{b+1}, v_b)$ to L_u on the inner boundary points (excluding the corners):

$$\begin{aligned} L_{b+1}(v_{b+1}, v_b) \approx & (v_b^{(1)} + v_{b+1}^{(7)} - 2v_{b+1}^{(0)}) + \frac{3}{4}(v_{b+1}^{(2)} + v_{b+1}^{(5)} - 2v_{b+1}^{(0)}) \\ & + \frac{1}{2}(v_{b+1}^{(6)} + v_{b+1}^{(8)} - v_{b+1}^{(2)} - v_{b+1}^{(4)}) \end{aligned} \quad (3.14)$$

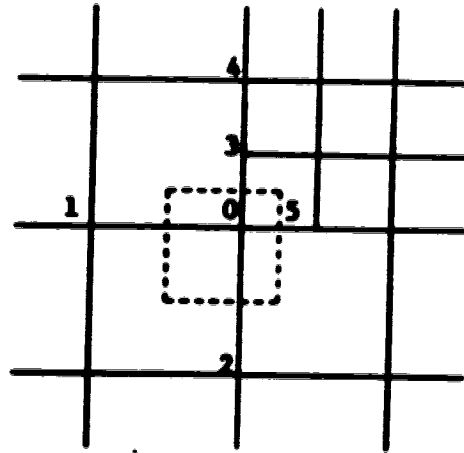
(Here, we have used $h_b = 2h_{b+1}$ for the simplification.)

For the corner points on the inner boundaries of the composite grid, there are several choices. First, similar to the treatment above, we may construct the finite volume as in Figure 3.5 (a). Then,

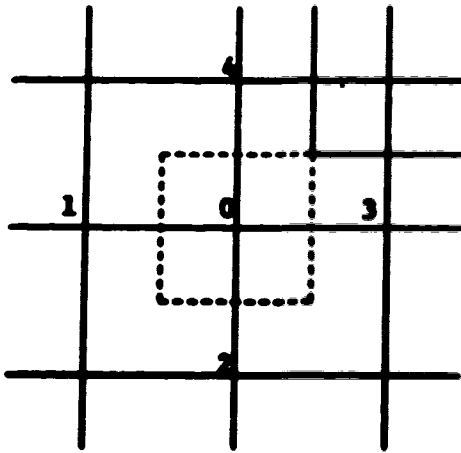
$$\begin{aligned} L_{b+1}(v_{b+1}, v_b) \approx & \frac{v_b^{(1)} - v_{b+1}^{(0)}}{h_b} \cdot h_b + \frac{v_b^{(2)} - v_{b+1}^{(0)}}{h_b} \cdot h_b \\ & + \frac{v_{b+1}^{(7)} - v_{b+1}^{(0)}}{2h_{b+1}} \cdot (\frac{h_b}{2} + \frac{h_{b+1}}{2}) + \frac{v_{b+1}^{(8)} - v_{b+1}^{(2)}}{2h_{b+1}} \cdot \frac{h_{b+1}}{2} \\ & + \frac{v_{b+1}^{(6)} - v_{b+1}^{(0)}}{2h_{b+1}} \cdot \frac{h_{b+1}}{2} + \frac{v_{b+1}^{(5)} - v_{b+1}^{(4)}}{2h_{b+1}} \cdot (\frac{h_b}{2} + \frac{h_{b+1}}{2}) \\ = & (v_b^{(1)} + v_b^{(2)} - 2v_{b+1}^{(0)}) + \frac{3}{4}(v_{b+1}^{(2)} + v_{b+1}^{(7)} - 2v_{b+1}^{(0)}) \\ & + \frac{1}{4}(v_{b+1}^{(6)} + v_{b+1}^{(8)} - v_{b+1}^{(2)} - v_{b+1}^{(4)}) \end{aligned} \quad (3.15)$$



(a)



(b)



(c)

Figure 3.5
Finite volume method
for inner boundary corner points

Another choice, illustrated in Figure 3.5 (b), is:

$$\begin{aligned}
\bar{L}_{k+1}(v_{k+1}, v_k) &\approx \frac{v_k^{(1)} - v_{k+1}^{(0)}}{h_k} \cdot \left(\frac{h_k}{2} + \frac{h_{k+1}}{2}\right) + \frac{v_k^{(2)} - v_{k+1}^{(0)}}{h_k} \cdot \left(\frac{h_k}{2} + \frac{h_{k+1}}{2}\right) \\
&\quad + \frac{v_{k+1}^{(3)} - v_{k+1}^{(0)}}{h_{k+1}} \cdot \left(\frac{h_k}{2} + \frac{h_{k+1}}{2}\right) + \frac{v_{k+1}^{(4)} - v_{k+1}^{(0)}}{h_{k+1}} \cdot \left(\frac{h_k}{2} + \frac{h_{k+1}}{2}\right) \\
&= \frac{3}{4}(v_k^{(1)} + v_k^{(2)} - 2v_{k+1}^{(0)}) + \frac{3}{2}(v_{k+1}^{(3)} + v_{k+1}^{(4)} - 2v_{k+1}^{(0)}) \quad (3.16)
\end{aligned}$$

The third choice, shown in Figure 3.5 (c), is:

$$\begin{aligned}
\bar{L}_{k+1}(v_{k+1}, v_k) &\approx \frac{v_k^{(1)} - v_{k+1}^{(0)}}{h_k} \cdot h_k + \frac{v_k^{(2)} - v_{k+1}^{(0)}}{h_k} \cdot h_k \\
&\quad + \frac{v_{k+1}^{(3)} - v_{k+1}^{(0)}}{2h_{k+1}} \cdot 2h_{k+1} + \frac{v_{k+1}^{(4)} - v_{k+1}^{(0)}}{2h_{k+1}} \cdot 2h_{k+1} \\
&= v_k^{(1)} + v_k^{(2)} + v_{k+1}^{(3)} + v_{k+1}^{(4)} - 4v_{k+1}^{(0)} \quad (3.17)
\end{aligned}$$

Due to the complicated structure of the multigrid method, we have not provided a rigorous convergence proof for our local mesh refinement algorithm; however, as we will see in the next section, the numerical results indicate that the convergence characteristics of the standard multi-grid method are maintained, but with substantial savings in storage and a very significant reduction in CPU time. Nevertheless, due to the structure of the composite grid, the definition of a work unit should be modified to be the computational work of one sweep on the whole composite grid Ω_{com} , not on the finest grid Ω_M alone.

4. Numerical Results.

For the model problem (3.1), as we mentioned before, when α gets quite big, the variation of the solutions near the upper right hand corner will be more rapid. Therefore, the grids near that corner should be refined to a certain degree to get more accurate solutions within that region.

For multi-grid method purposes, we construct the first several grids in the uniform way. For our model problem, we let the first three grids be uniform, and the numbers of grid points are 3×3 , 5×5 , and 9×9 , respectively. Then, we refine the upper right hand corner twice to get another two finer local grids, 9×9 each. The composite grid Ω_{com} is shown in *Figure 4.1*. The coarsest grid shown here is the third grid and the finest one is the fifth grid.

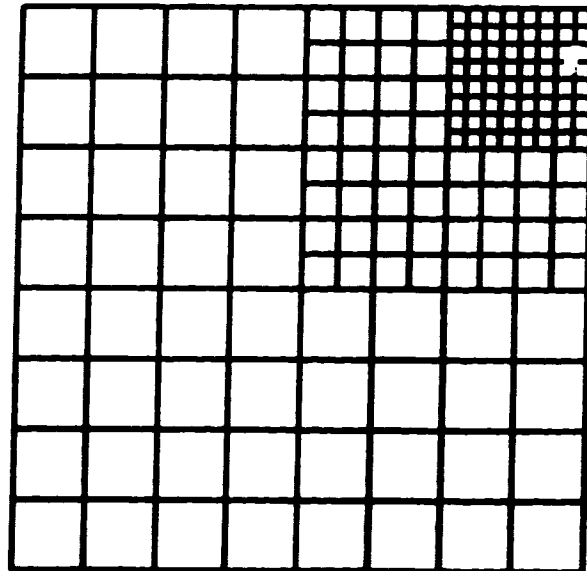


Figure 4.1

A composite grid Ω_{com} for model problem

For $\alpha = 100$, by using injection (2.20) as the restriction R_h^H and bi-linear interpolation (2.22) as the prolongation I_H^h , we have the results shown in Table 4.1, and the corresponding graphs of solution are shown in Figure 4.2. Moreover, for the convenience of comparison, we give the results derived from applying uniform multi-grid method onto the same problem in Table 4.2.

Thus, from Table 4.1 and Table 4.2, it is obvious that our local mesh refinement method is very successful in treating problems where there is rapid variation in certain regions of the domain: the mesh-size independent convergence property has been retained and the higher accuracy has been achieved in the local region as well as the result derived from using the uniform multi-grid method. Moreover, the significant achievement here is the great saving in CPU time. By comparing the time spent on grids 4 and 5 in both tables, we found that the ratio of the time spent on the same finest grid between uniform and composite grids approximately equals the ratio of the areas both grids cover.

In section 3.4, we have discussed the three possible local corner point treatments. The results above have been obtained by using the local boundary treatment choice (b). If we use choice (a) or (c), then we may get the following result of the corresponding absolute errors: $0.435296e-2$ and $0.435372e-2$. The small difference among these three choices is probably due to the small influence of these corner points (if the local region has sufficient grid points).

Finally, we point out that because the coefficients of our model problem are very smooth, numerical results also show that there is no obvious difference between the choice of injection or full-weighting as the restriction operator.

No. of Grid Points	No. of Grid Level	No. of Iterations	CPU (millisec.)	Error $ v_h - u^* $
3×3	1	2	24	-
5×5	2	3	121	-
9×9	3	4	654	$0.700677e-1$
9×9	4	4	829	$0.100698e-1$
9×9	5	4	1017	$0.434965e-2$

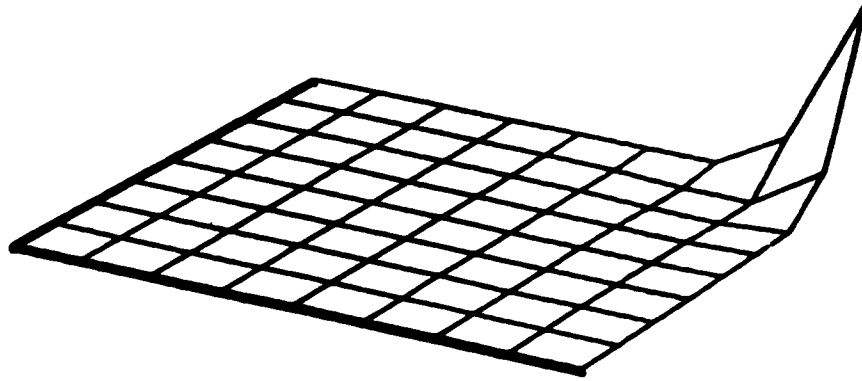
Table 4.1

Local mesh refinement algorithm on model problem ($\alpha = 10^3$)

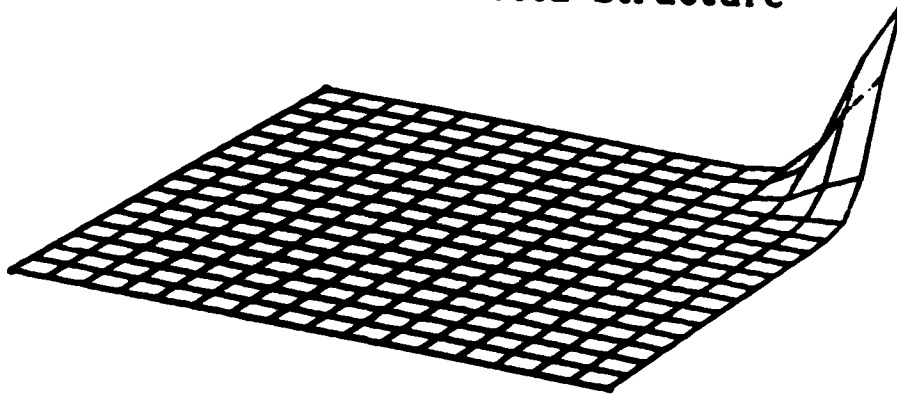
No. of Grid Points	No. of Grid Level	No. of Iterations	CPU (millisec.)	Error $ v_h - u^* $
3×3	1	2	24	-
5×5	2	3	121	-
9×9	3	4	654	$0.700677e-1$
17×17	4	4	3485	$0.100697e-1$
33×33	5	4	15061	$0.430037e-2$

Table 4.2

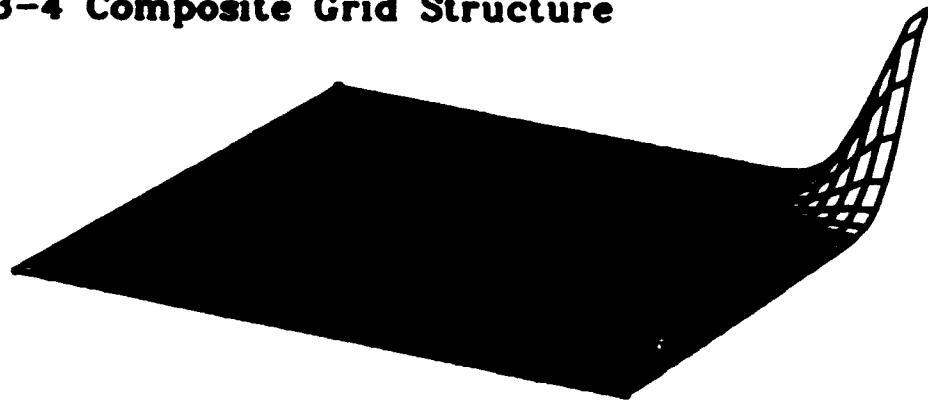
Uniform multi-grid method on model problem ($\alpha = 10^3$)



(a) 3-3 Uniform Grid Structure



(b) 3-4 Composite Grid Structure



(c) 3-5 Composite Grid Structure

Figure 4.8

Bibliography

- [1] Auzinger and H. J. Stetter, *Defect correction and multi-grid iterations*, in *Multi-grid methods*, Proc. conference held at Köln-Porz, November 23-27, 1981, W. Hackbusch and U. Trottenberg, eds., *Lecture Notes in Mathematics* 900, Springer-Verlag, Berlin, 1982, pp.327-351.
- [2] D. Bai and A. Brandt, *Local mesh refinement multi-level techniques*, *SIAM J. Sci. Stat. Comput.* 8(1987), pp.109-134.
- [3] R. E. Bank, A. H. Sherman and A. Weiser, *Refinement algorithms and data structures for regular local mesh refinement*, in *Scientific Computing*, R. S. Stepleman, ed., North-Holland, Amsterdam, 1983.
- [4] D. Braess, *The convergence rate of a multi-grid method with Gauss-Seidel relaxation for the Poisson equation (revised)*, *Math. Comp.*, 42(1984), pp.505-519.
- [5] D. Braess, W. Hackbusch and U. trottenberg, eds., *Advances in multi-grid methods*, Proc. conference held in Oberwolfach, December 8-12, 1984, *Notes on Numerical Fluid Mechanics*, Vol.11, Vieweg, Braunschweig, 1984.
- [6] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, *Math. Comp.*, 31(1977), pp.339-380.
- [7] A. Brandt, *Multi-level adaptive techniques (MLAT) for partial differential equations: ideas and software*, in *Mathematical Software III*, J. R. Rice, ed., Academic Press, New York, 1977, pp.277-312.
- [8] R. Ewing, S. F. McCormick and J. Thomas, *The fast adaptive composite grid method for solving differential boundary value problems*, Proc. Fifth ASCE-EMD Speciality Conference, August 1984.

- [9] H. Forster and K. Witsch, *Multi-grid software for the solution of elliptic problems on rectangular domains: MG00 (Release 1)*, in *Multi-grid methods*, Proc. conference held at Köln-Porz, November 23-27, 1981, W. Hackbusch and U. Trottenberg, eds., *Lecture Notes in Mathematics* 900, Springer-Verlag, Berlin, 1982.
- [10] L. Fuchs, *A Newton-multi-grid method for the solution of non-linear partial differential equations*, in *Boundary and Interior Layers — Computational and Asymptotic Methods*, J. J. H. Miller, ed., *Books Press*, Dublin, 1980, pp.291-296.
- [11] W. Hackbusch, *Multi-grid methods and applications*, *Springer Series in Comp. Math.* 4, Springer-Verlag, Berlin, 1985.
- [12] W. Hackbusch, *Local defect correction method and domain decomposition techniques*, in *Computing, suppl.5, Defect Correction Methods*, K. Böhmer and H. J. Stetter, eds., Springer-Verlag, Wien, 1984, pp.89-113.
- [13] W. Hackbusch, *Convergence of multi-grid iterations applied to difference equations*, *Math. Comp.* 34(1980), pp.425-440.
- [14] D. A. H. Jacobs, *Some iterative methods: past, present, future*, *IEEE Colloquium on Numerical Solution Techniques for Large Sparse Systems of Equations*, London, November 1983.
- [15] D. C. Jasperen, *Multi-grid methods for partial differential equations*, in *Studies in Numerical Analysis*, G. Golub, ed., *MAA*, 1984.
- [16] C. Q. Liu and S. F. McCormick, *Multi-grid elliptic grid generation and the fast adaptive composite grid method for solving transonic potential flow equations*, in *Multigrid methods, theory, applications and supercomputing*, S. F. McCormick. ed., *Marcel Dekker*, New York, 1988, pp.365-387.

- [17] S. F. McCormick, *Fast adaptive grid (FAC) methods: Theory for the variational case*, in *Computing, Suppl.5, Defect Correction Methods*, K. Böhm and H. J. Stetter, eds., Springer-Verlag, Vienna, 1984.
- [18] J. R. van Rosendale, *Algorithms and data structures for adaptive multi-grid elliptic solvers*, *Appl. Math. Comp.*, 13, Proc. Internat. Multi-grid Conference, April 6-8, 1983, Copper Mountain, Co., S. F. McCormick and U. Trottenberg, eds., North-Holland, Amsterdam, 1983, pp.453-470.
- [19] G. J. Shaw and S. Sivaloganathan, *On the smoothing properties of the simple pressure-correction algorithm*, *Internat. J. Numer. Methods Fluids*, Vol.8, No.4, 1988, pp.441-461.
- [20] S. Sivaloganathan and G. J. Shaw, *A multi-grid method for recirculating flows*, *Internat. J. Numer. Methods Fluids*, Vol.7, 1987, pp.417-440.
- [21] D. M. Young and R. T. Gregory, *A survey of numerical mathematics*, Vol.182, Addison-Wesley, 1972.
- [22] T. A. Zang, Y. S. Wong and M. Y. Hussaini, *Spectral multi-grid methods for elliptic equations*, *J. Comput. Phys.*, 49(1982), pp.485-501.
- [23] T. A. Zang, Y. S. Wong and M. Y. Hussaini, *Spectral multi-grid methods for elliptic equations II*, NASA-CR-172131, ICASE, NASA Langley Research Center, 1983.

Appendix: List of a FORTRAN77 Program for LMRA

100

14-00000

```

123 C      (Get PFC, post- and pre-post- relaxation numbers)
124 C      READ=7
125 C      UPDS=2
126 C      UPST=1
127 C      CORRECT=100
128
129 C      (Get GPC relaxation parameter GPCG):
130 C      DO 100 IPDS=1,UPDS
131 C      W(IPDS)=1.000
132 C      CONTINUE
133
134 C      (Get relaxation category)
135 C      (Note: In this program, relaxation is bilinear)
136 C      READ=3
137
138 C      (Get u-cube style)
139 C      READ=2
140
141 C      (Get corner type)
142 C      READ=1
143
144 C      GO TO 300
145
146 C
147 C      GOBT17
148
149 C      .....
150 C      .....
151 C      READ: line of relaxation method used as a parameter
152 C      1 - Point Gauss-Seidel
153 C      2 - Symmetric Point Gauss-Seidel
154 C      3 - Jacobi
155 C      4 - u-Line Gauss-Seidel
156 C      5 - v-Line Gauss-Seidel
157 C      6 - Alternating Line Gauss-Seidel
158 C      7 - Alternating Symmetric Line Gauss-Seidel
159 C      8 - u-Line Gauss
160 C      9 - v-Line Gauss
161 C      10 - Alternating Line Gauss
162
163 C      READ(10,*) READ
164
165 C      .....
166 C      READ: max no of pre-relaxation sweeps
167 C      READ(10,*) UPDS
168 C      .....
169 C      UPST: max no of post-relaxation sweeps
170 C      READ(10,*) UPST
171 C      .....
172 C      CORRECT: max no of relaxations on correct mesh
173 C      READ(10,*) CORRECT
174 C      .....
175 C      W: relaxation parameter GPCG for each variable in system
176 C      DO 100 IPDS=1,UPDS
177 C      W(IPDS)=1.000
178 C      CONTINUE
179
180 C      (Get PFC, post- and pre-post- relaxation numbers)
181 C      READ=7
182 C      UPDS=2
183 C      UPST=1
184 C      CORRECT=100
185
186 C      (Get GPC relaxation parameter GPCG):
187 C      DO 100 IPDS=1,UPDS
188 C      W(IPDS)=1.000
189 C      CONTINUE
190
191 C      (Get relaxation category)
192 C      (Note: In this program, relaxation is bilinear)
193 C      READ=3
194
195 C      (Get u-cube style)
196 C      READ=2
197
198 C      (Get corner type)
199 C      READ=1
200
201 C      GO TO 300
202
203 C
204 C      GOBT17
205
206 C      .....
207 C      .....
208 C      READ: line of relaxation method used
209 C      1 - Point Gauss-Seidel
210 C      2 - Symmetric Point Gauss-Seidel
211 C      3 - Jacobi
212 C      4 - u-Line Gauss-Seidel
213 C      5 - v-Line Gauss-Seidel
214 C      6 - Alternating Line Gauss-Seidel
215 C      7 - Alternating Symmetric Line Gauss-Seidel
216 C      8 - u-Line Gauss
217 C      9 - v-Line Gauss
218 C      10 - Alternating Line Gauss
219
220 C      READ(10,*) READ
221
222 C      .....
223 C      READ: no of sweeps per relaxation
224 C      1 - u-cube
225 C      2 - v-cube
226
227 C      READ(10,*) READ
228
229 C      .....
230 C      READ: corner value for type (1,2, or 3)
231 C      READ(10,*) READ
232 C      .....
233
234 C      (Get GPC relaxation parameter GPCG for each variable in system)
235 C      DO 100 IPDS=1,UPDS
236 C      W(IPDS)=1.000
237 C      CONTINUE
238
239 C      (Get PFC, post- and pre-post- relaxation numbers)
240 C      READ=7
241 C      UPDS=2
242 C      UPST=1
243 C      CORRECT=100
244
245 C      (Get GPC relaxation parameter GPCG):
246 C      DO 100 IPDS=1,UPDS
247 C      W(IPDS)=1.000
248 C      CONTINUE
249
250 C      (Get relaxation category)
251 C      (Note: In this program, relaxation is bilinear)
252 C      READ=3
253
254 C      (Get u-cube style)
255 C      READ=2
256
257 C      (Get corner type)
258 C      READ=1
259
260 C      GO TO 300
261
262 C
263 C      GOBT17
264
265 C      .....
266 C      .....
267 C      READ: line of relaxation method used
268 C      1 - Point Gauss-Seidel
269 C      2 - Symmetric Point Gauss-Seidel
270 C      3 - Jacobi
271 C      4 - u-Line Gauss-Seidel
272 C      5 - v-Line Gauss-Seidel
273 C      6 - Alternating Line Gauss-Seidel
274 C      7 - Alternating Symmetric Line Gauss-Seidel
275 C      8 - u-Line Gauss
276 C      9 - v-Line Gauss
277 C      10 - Alternating Line Gauss
278
279 C      READ(10,*) READ
280
281 C      .....
282 C      READ: no of sweeps per relaxation
283 C      1 - u-cube
284 C      2 - v-cube
285
286 C      READ(10,*) READ
287
288 C      .....
289 C      READ: corner value for type (1,2, or 3)
290 C      READ(10,*) READ
291 C      .....
292
293 C      (Get GPC relaxation parameter GPCG for each variable in system)
294 C      DO 100 IPDS=1,UPDS
295 C      W(IPDS)=1.000
296 C      CONTINUE
297
298 C      (Get PFC, post- and pre-post- relaxation numbers)
299 C      READ=7
300 C      UPDS=2
301 C      UPST=1
302 C      CORRECT=100
303
304 C      (Get GPC relaxation parameter GPCG):
305 C      DO 100 IPDS=1,UPDS
306 C      W(IPDS)=1.000
307 C      CONTINUE
308
309 C      (Get relaxation category)
310 C      (Note: In this program, relaxation is bilinear)
311 C      READ=3
312
313 C      (Get u-cube style)
314 C      READ=2
315
316 C      (Get corner type)
317 C      READ=1
318
319 C      GO TO 300
320
321 C
322 C      GOBT17
323
324 C      .....
325 C      .....
326 C      READ: line of relaxation method used
327 C      1 - Point Gauss-Seidel
328 C      2 - Symmetric Point Gauss-Seidel
329 C      3 - Jacobi
330 C      4 - u-Line Gauss-Seidel
331 C      5 - v-Line Gauss-Seidel
332 C      6 - Alternating Line Gauss-Seidel
333 C      7 - Alternating Symmetric Line Gauss-Seidel
334 C      8 - u-Line Gauss
335 C      9 - v-Line Gauss
336 C      10 - Alternating Line Gauss
337
338 C      READ(10,*) READ
339
340 C      .....
341 C      READ: no of sweeps per relaxation
342 C      1 - u-cube
343 C      2 - v-cube
344
345 C      READ(10,*) READ
346
347 C      .....
348 C      READ: corner value for type (1,2, or 3)
349 C      READ(10,*) READ
350 C      .....
351
352 C      (Get GPC relaxation parameter GPCG for each variable in system)
353 C      DO 100 IPDS=1,UPDS
354 C      W(IPDS)=1.000
355 C      CONTINUE
356
357 C      (Get PFC, post- and pre-post- relaxation numbers)
358 C      READ=7
359 C      UPDS=2
360 C      UPST=1
361 C      CORRECT=100
362
363 C      (Get GPC relaxation parameter GPCG):
364 C      DO 100 IPDS=1,UPDS
365 C      W(IPDS)=1.000
366 C      CONTINUE
367
368 C      (Get relaxation category)
369 C      (Note: In this program, relaxation is bilinear)
370 C      READ=3
371
372 C      (Get u-cube style)
373 C      READ=2
374
375 C      (Get corner type)
376 C      READ=1
377
378 C      GO TO 300
379
380 C
381 C      GOBT17
382
383 C      .....
384 C      .....
385 C      READ: line of relaxation method used
386 C      1 - Point Gauss-Seidel
387 C      2 - Symmetric Point Gauss-Seidel
388 C      3 - Jacobi
389 C      4 - u-Line Gauss-Seidel
390 C      5 - v-Line Gauss-Seidel
391 C      6 - Alternating Line Gauss-Seidel
392 C      7 - Alternating Symmetric Line Gauss-Seidel
393 C      8 - u-Line Gauss
394 C      9 - v-Line Gauss
395 C      10 - Alternating Line Gauss
396
397 C      READ(10,*) READ
398
399 C      .....
400 C      READ: no of sweeps per relaxation
401 C      1 - u-cube
402 C      2 - v-cube
403
404 C      READ(10,*) READ
405
406 C      .....
407 C      READ: corner value for type (1,2, or 3)
408 C      READ(10,*) READ
409 C      .....
410
411 C      (Get GPC relaxation parameter GPCG for each variable in system)
412 C      DO 100 IPDS=1,UPDS
413 C      W(IPDS)=1.000
414 C      CONTINUE
415
416 C      (Get PFC, post- and pre-post- relaxation numbers)
417 C      READ=7
418 C      UPDS=2
419 C      UPST=1
420 C      CORRECT=100
421
422 C      (Get GPC relaxation parameter GPCG):
423 C      DO 100 IPDS=1,UPDS
424 C      W(IPDS)=1.000
425 C      CONTINUE
426
427 C      (Get relaxation category)
428 C      (Note: In this program, relaxation is bilinear)
429 C      READ=3
430
431 C      (Get u-cube style)
432 C      READ=2
433
434 C      (Get corner type)
435 C      READ=1
436
437 C      GO TO 300
438
439 C
440 C      GOBT17
441
442 C      .....
443 C      .....
444 C      READ: line of relaxation method used
445 C      1 - Point Gauss-Seidel
446 C      2 - Symmetric Point Gauss-Seidel
447 C      3 - Jacobi
448 C      4 - u-Line Gauss-Seidel
449 C      5 - v-Line Gauss-Seidel
450 C      6 - Alternating Line Gauss-Seidel
451 C      7 - Alternating Symmetric Line Gauss-Seidel
452 C      8 - u-Line Gauss
453 C      9 - v-Line Gauss
454 C      10 - Alternating Line Gauss
455
456 C      READ(10,*) READ
457
458 C      .....
459 C      READ: no of sweeps per relaxation
460 C      1 - u-cube
461 C      2 - v-cube
462
463 C      READ(10,*) READ
464
465 C      .....
466 C      READ: corner value for type (1,2, or 3)
467 C      READ(10,*) READ
468 C      .....
469
470 C      (Get GPC relaxation parameter GPCG for each variable in system)
471 C      DO 100 IPDS=1,UPDS
472 C      W(IPDS)=1.000
473 C      CONTINUE
474
475 C      (Get PFC, post- and pre-post- relaxation numbers)
476 C      READ=7
477 C      UPDS=2
478 C      UPST=1
479 C      CORRECT=100
480
481 C      (Get GPC relaxation parameter GPCG):
482 C      DO 100 IPDS=1,UPDS
483 C      W(IPDS)=1.000
484 C      CONTINUE
485
486 C      (Get relaxation category)
487 C      (Note: In this program, relaxation is bilinear)
488 C      READ=3
489
490 C      (Get u-cube style)
491 C      READ=2
492
493 C      (Get corner type)
494 C      READ=1
495
496 C      GO TO 300
497
498 C
499 C      GOBT17
500
501 C      .....
502 C      .....
503 C      READ: line of relaxation method used
504 C      1 - Point Gauss-Seidel
505 C      2 - Symmetric Point Gauss-Seidel
506 C      3 - Jacobi
507 C      4 - u-Line Gauss-Seidel
508 C      5 - v-Line Gauss-Seidel
509 C      6 - Alternating Line Gauss-Seidel
51
```


LOGGING OF LINES FOR 02-10-25 ON APR 12, 1960 FOR SET-POINTS ON SOLVENTS

[illegible]

SECRET

```

000 C SUBROUTINE SUB1:
001 IMPLICIT REAL*8 (A-H,O-Z)
002 COMMON C100000,Z1,Z2101,Z3101,Z4101,Z5101
003 COMMON /PARMS/ Z101,Z1101,Z12101,Z2101,Z21101,Z21201
004 COMMON /RPGS/ R0000,R0001,R0002,R0003
005 COMMON /PARMS/ EPS
006
007 C .....
008 C Define boundary on the whole region
009 C .....
010 C
011 SUB1:
012 EPS=1.000/EPG
013 M=1.000/EPLOSTIN-1
014
015 C
016 DO 100 J=J011,Z101
017 DO 100 I=I011,Z101
018 Z=Z101+I-J-1.000-1
019 DO 100 IP00=1,EPG00
020 IF 11.00.11 THEN
021 GO TO 30
022
023 SUB1P
024 IF 12.00.11 THEN
025 GO TO 30
026
027 SUB1P
028 IF 13.00.11 THEN
029 GO TO 30
030
031 SUB1P
032 IF 14.00.11 THEN
033 GO TO 30
034
035 SUB1P
036 IF 15.00.11 THEN
037 GO TO 30
038
039 SUB1P
040 IF 16.00.11 THEN
041 GO TO 30
042
043 SUB1P
044 IF 17.00.11 THEN
045 GO TO 30
046
047 SUB1P
048 IF 18.00.11 THEN
049 GO TO 30
050
051 SUB1P
052 IF 19.00.11 THEN
053 GO TO 30
054
055 SUB1P
056 IF 20.00.11 THEN
057 GO TO 30
058
059 SUB1P
060 IF 21.00.11 THEN
061 GO TO 30
062
063 SUB1P
064 IF 22.00.11 THEN
065 GO TO 30
066
067 SUB1P
068 IF 23.00.11 THEN
069 GO TO 30
070
071 SUB1P
072 IF 24.00.11 THEN
073 GO TO 30
074
075 SUB1P
076 IF 25.00.11 THEN
077 GO TO 30
078
079 SUB1P
080 IF 26.00.11 THEN
081 GO TO 30
082
083 SUB1P
084 IF 27.00.11 THEN
085 GO TO 30
086
087 SUB1P
088 IF 28.00.11 THEN
089 GO TO 30
090
091 SUB1P
092 IF 29.00.11 THEN
093 GO TO 30
094
095 SUB1P
096 IF 30.00.11 THEN
097 GO TO 30
098
099 SUB1P
100 IF 31.00.11 THEN
101 GO TO 30
102
103 SUB1P
104 IF 32.00.11 THEN
105 GO TO 30
106
107 SUB1P
108 IF 33.00.11 THEN
109 GO TO 30
110
111 SUB1P
112 IF 34.00.11 THEN
113 GO TO 30
114
115 SUB1P
116 IF 35.00.11 THEN
117 GO TO 30
118
119 SUB1P
120 IF 36.00.11 THEN
121 GO TO 30
122
123 SUB1P
124 IF 37.00.11 THEN
125 GO TO 30
126
127 SUB1P
128 IF 38.00.11 THEN
129 GO TO 30
130
131 SUB1P
132 IF 39.00.11 THEN
133 GO TO 30
134
135 SUB1P
136 IF 40.00.11 THEN
137 GO TO 30
138
139 SUB1P
140 IF 41.00.11 THEN
141 GO TO 30
142
143 SUB1P
144 IF 42.00.11 THEN
145 GO TO 30
146
147 SUB1P
148 IF 43.00.11 THEN
149 GO TO 30
150
151 SUB1P
152 IF 44.00.11 THEN
153 GO TO 30
154
155 SUB1P
156 IF 45.00.11 THEN
157 GO TO 30
158
159 SUB1P
160 IF 46.00.11 THEN
161 GO TO 30
162
163 SUB1P
164 IF 47.00.11 THEN
165 GO TO 30
166
167 SUB1P
168 IF 48.00.11 THEN
169 GO TO 30
170
171 SUB1P
172 IF 49.00.11 THEN
173 GO TO 30
174
175 SUB1P
176 IF 50.00.11 THEN
177 GO TO 30
178
179 SUB1P
180 IF 51.00.11 THEN
181 GO TO 30
182
183 SUB1P
184 IF 52.00.11 THEN
185 GO TO 30
186
187 SUB1P
188 IF 53.00.11 THEN
189 GO TO 30
190
191 SUB1P
192 IF 54.00.11 THEN
193 GO TO 30
194
195 SUB1P
196 IF 55.00.11 THEN
197 GO TO 30
198
199 SUB1P
200 IF 56.00.11 THEN
201 GO TO 30
202
203 SUB1P
204 IF 57.00.11 THEN
205 GO TO 30
206
207 SUB1P
208 IF 58.00.11 THEN
209 GO TO 30
210
211 SUB1P
212 IF 59.00.11 THEN
213 GO TO 30
214
215 SUB1P
216 IF 60.00.11 THEN
217 GO TO 30
218
219 SUB1P
220 IF 61.00.11 THEN
221 GO TO 30
222
223 SUB1P
224 IF 62.00.11 THEN
225 GO TO 30
226
227 SUB1P
228 IF 63.00.11 THEN
229 GO TO 30
230
231 SUB1P
232 IF 64.00.11 THEN
233 GO TO 30
234
235 SUB1P
236 IF 65.00.11 THEN
237 GO TO 30
238
239 SUB1P
240 IF 66.00.11 THEN
241 GO TO 30
242
243 SUB1P
244 IF 67.00.11 THEN
245 GO TO 30
246
247 SUB1P
248 IF 68.00.11 THEN
249 GO TO 30
250
251 SUB1P
252 IF 69.00.11 THEN
253 GO TO 30
254
255 SUB1P
256 IF 70.00.11 THEN
257 GO TO 30
258
259 SUB1P
260 IF 71.00.11 THEN
261 GO TO 30
262
263 SUB1P
264 IF 72.00.11 THEN
265 GO TO 30
266
267 SUB1P
268 IF 73.00.11 THEN
269 GO TO 30
270
271 SUB1P
272 IF 74.00.11 THEN
273 GO TO 30
274
275 SUB1P
276 IF 75.00.11 THEN
277 GO TO 30
278
279 SUB1P
280 IF 76.00.11 THEN
281 GO TO 30
282
283 SUB1P
284 IF 77.00.11 THEN
285 GO TO 30
286
287 SUB1P
288 IF 78.00.11 THEN
289 GO TO 30
290
291 SUB1P
292 IF 79.00.11 THEN
293 GO TO 30
294
295 SUB1P
296 IF 80.00.11 THEN
297 GO TO 30
298
299 SUB1P
300 IF 81.00.11 THEN
301 GO TO 30
302
303 SUB1P
304 IF 82.00.11 THEN
305 GO TO 30
306
307 SUB1P
308 IF 83.00.11 THEN
309 GO TO 30
310
311 SUB1P
312 IF 84.00.11 THEN
313 GO TO 30
314
315 SUB1P
316 IF 85.00.11 THEN
317 GO TO 30
318
319 SUB1P
320 IF 86.00.11 THEN
321 GO TO 30
322
323 SUB1P
324 IF 87.00.11 THEN
325 GO TO 30
326
327 SUB1P
328 IF 88.00.11 THEN
329 GO TO 30
330
331 SUB1P
332 IF 89.00.11 THEN
333 GO TO 30
334
335 SUB1P
336 IF 90.00.11 THEN
337 GO TO 30
338
339 SUB1P
340 IF 91.00.11 THEN
341 GO TO 30
342
343 SUB1P
344 IF 92.00.11 THEN
345 GO TO 30
346
347 SUB1P
348 IF 93.00.11 THEN
349 GO TO 30
350
351 SUB1P
352 IF 94.00.11 THEN
353 GO TO 30
354
355 SUB1P
356 IF 95.00.11 THEN
357 GO TO 30
358
359 SUB1P
360 IF 96.00.11 THEN
361 GO TO 30
362
363 SUB1P
364 IF 97.00.11 THEN
365 GO TO 30
366
367 SUB1P
368 IF 98.00.11 THEN
369 GO TO 30
370
371 SUB1P
372 IF 99.00.11 THEN
373 GO TO 30
374
375 SUB1P
376 IF 100.00.11 THEN
377 GO TO 30
378
379 SUB1P
380 IF 101.00.11 THEN
381 GO TO 30
382
383 SUB1P
384 IF 102.00.11 THEN
385 GO TO 30
386
387 SUB1P
388 IF 103.00.11 THEN
389 GO TO 30
390
391 SUB1P
392 IF 104.00.11 THEN
393 GO TO 30
394
395 SUB1P
396 IF 105.00.11 THEN
397 GO TO 30
398
399 SUB1P
400 IF 106.00.11 THEN
401 GO TO 30
402
403 SUB1P
404 IF 107.00.11 THEN
405 GO TO 30
406
407 SUB1P
408 IF 108.00.11 THEN
409 GO TO 30
410
411 SUB1P
412 IF 109.00.11 THEN
413 GO TO 30
414
415 SUB1P
416 IF 110.00.11 THEN
417 GO TO 30
418
419 SUB1P
420 IF 111.00.11 THEN
421 GO TO 30
422
423 SUB1P
424 IF 112.00.11 THEN
425 GO TO 30
426
427 SUB1P
428 IF 113.00.11 THEN
429 GO TO 30
430
431 SUB1P
432 IF 114.00.11 THEN
433 GO TO 30
434
435 SUB1P
436 IF 115.00.11 THEN
437 GO TO 30
438
439 SUB1P
440 IF 116.00.11 THEN
441 GO TO 30
442
443 SUB1P
444 IF 117.00.11 THEN
445 GO TO 30
446
447 SUB1P
448 IF 118.00.11 THEN
449 GO TO 30
450
45
```

Effects of Age

```

001      CALL=OPLOCAT(11+1)
002      GO TO 1
003
004      CALL=OPLOCAT(11+1)
005
006      IF (NEND.EQ.1) THEN
007          CALL=OPLOCAT(11+1)
008      ELSE
009          CALL=OPLOCAT(11+1)
010      ENDIF
011
012      GO TO 1
013
014      CALL=OPLOCAT(11+1)
015
016      IF (NEND.EQ.1) THEN
017          CALL=OPLOCAT(11+1)
018      ELSE
019          CALL=OPLOCAT(11+1)
020      ENDIF
021
022      GO TO 1
023
024      CALL=OPLOCAT(11+1)
025
026      IF (NEND.EQ.1) THEN
027          CALL=OPLOCAT(11+1)
028      ELSE
029          CALL=OPLOCAT(11+1)
030      ENDIF
031
032      GO TO 1
033
034      CALL=OPLOCAT(11+1)
035
036      IF (NEND.EQ.1) THEN
037          CALL=OPLOCAT(11+1)
038      ELSE
039          CALL=OPLOCAT(11+1)
040      ENDIF
041
042      GO TO 1
043
044      CALL=OPLOCAT(11+1)
045
046      IF (NEND.EQ.1) THEN
047          CALL=OPLOCAT(11+1)
048      ELSE
049          CALL=OPLOCAT(11+1)
050      ENDIF
051
052      GO TO 1
053
054      CALL=OPLOCAT(11+1)
055
056      IF (NEND.EQ.1) THEN
057          CALL=OPLOCAT(11+1)
058      ELSE
059          CALL=OPLOCAT(11+1)
060      ENDIF
061
062      GO TO 1
063
064      CALL=OPLOCAT(11+1)
065
066      IF (NEND.EQ.1) THEN
067          CALL=OPLOCAT(11+1)
068      ELSE
069          CALL=OPLOCAT(11+1)
070      ENDIF
071
072      GO TO 1
073
074      CALL=OPLOCAT(11+1)
075
076      IF (NEND.EQ.1) THEN
077          CALL=OPLOCAT(11+1)
078      ELSE
079          CALL=OPLOCAT(11+1)
080      ENDIF
081
082      GO TO 1
083
084      CALL=OPLOCAT(11+1)
085
086      IF (NEND.EQ.1) THEN
087          CALL=OPLOCAT(11+1)
088      ELSE
089          CALL=OPLOCAT(11+1)
090      ENDIF
091
092      GO TO 1
093
094      CALL=OPLOCAT(11+1)
095
096      IF (NEND.EQ.1) THEN
097          CALL=OPLOCAT(11+1)
098      ELSE
099          CALL=OPLOCAT(11+1)
100      ENDIF
101
102      GO TO 1
103
104      CALL=OPLOCAT(11+1)
105
106      IF (NEND.EQ.1) THEN
107          CALL=OPLOCAT(11+1)
108      ELSE
109          CALL=OPLOCAT(11+1)
110      ENDIF
111
112      GO TO 1
113
114      CALL=OPLOCAT(11+1)
115
116      IF (NEND.EQ.1) THEN
117          CALL=OPLOCAT(11+1)
118      ELSE
119          CALL=OPLOCAT(11+1)
120      ENDIF
121
122      GO TO 1
123
124      CALL=OPLOCAT(11+1)
125
126      IF (NEND.EQ.1) THEN
127          CALL=OPLOCAT(11+1)
128      ELSE
129          CALL=OPLOCAT(11+1)
130      ENDIF
131
132      GO TO 1
133
134      CALL=OPLOCAT(11+1)
135
136      IF (NEND.EQ.1) THEN
137          CALL=OPLOCAT(11+1)
138      ELSE
139          CALL=OPLOCAT(11+1)
140      ENDIF
141
142      GO TO 1
143
144      CALL=OPLOCAT(11+1)
145
146      IF (NEND.EQ.1) THEN
147          CALL=OPLOCAT(11+1)
148      ELSE
149          CALL=OPLOCAT(11+1)
150      ENDIF
151
152      GO TO 1
153
154      CALL=OPLOCAT(11+1)
155
156      IF (NEND.EQ.1) THEN
157          CALL=OPLOCAT(11+1)
158      ELSE
159          CALL=OPLOCAT(11+1)
160      ENDIF
161
162      GO TO 1
163
164      CALL=OPLOCAT(11+1)
165
166      IF (NEND.EQ.1) THEN
167          CALL=OPLOCAT(11+1)
168      ELSE
169          CALL=OPLOCAT(11+1)
170      ENDIF
171
172      GO TO 1
173
174      CALL=OPLOCAT(11+1)
175
176      IF (NEND.EQ.1) THEN
177          CALL=OPLOCAT(11+1)
178      ELSE
179          CALL=OPLOCAT(11+1)
180      ENDIF
181
182      GO TO 1
183
184      CALL=OPLOCAT(11+1)
185
186      IF (NEND.EQ.1) THEN
187          CALL=OPLOCAT(11+1)
188      ELSE
189          CALL=OPLOCAT(11+1)
190      ENDIF
191
192      GO TO 1
193
194      CALL=OPLOCAT(11+1)
195
196      IF (NEND.EQ.1) THEN
197          CALL=OPLOCAT(11+1)
198      ELSE
199          CALL=OPLOCAT(11+1)
200      ENDIF
201
202      GO TO 1
203
204      CALL=OPLOCAT(11+1)
205
206      IF (NEND.EQ.1) THEN
207          CALL=OPLOCAT(11+1)
208      ELSE
209          CALL=OPLOCAT(11+1)
210      ENDIF
211
212      GO TO 1
213
214      CALL=OPLOCAT(11+1)
215
216      IF (NEND.EQ.1) THEN
217          CALL=OPLOCAT(11+1)
218      ELSE
219          CALL=OPLOCAT(11+1)
220      ENDIF
221
222      GO TO 1
223
224      CALL=OPLOCAT(11+1)
225
226      IF (NEND.EQ.1) THEN
227          CALL=OPLOCAT(11+1)
228      ELSE
229          CALL=OPLOCAT(11+1)
230      ENDIF
231
232      GO TO 1
233
234      CALL=OPLOCAT(11+1)
235
236      IF (NEND.EQ.1) THEN
237          CALL=OPLOCAT(11+1)
238      ELSE
239          CALL=OPLOCAT(11+1)
240      ENDIF
241
242      GO TO 1
243
244      CALL=OPLOCAT(11+1)
245
246      IF (NEND.EQ.1) THEN
247          CALL=OPLOCAT(11+1)
248      ELSE
249          CALL=OPLOCAT(11+1)
250      ENDIF
251
252      GO TO 1
253
254      CALL=OPLOCAT(11+1)
255
256      IF (NEND.EQ.1) THEN
257          CALL=OPLOCAT(11+1)
258      ELSE
259          CALL=OPLOCAT(11+1)
260      ENDIF
261
262      GO TO 1
263
264      CALL=OPLOCAT(11+1)
265
266      IF (NEND.EQ.1) THEN
267          CALL=OPLOCAT(11+1)
268      ELSE
269          CALL=OPLOCAT(11+1)
270      ENDIF
271
272      GO TO 1
273
274      CALL=OPLOCAT(11+1)
275
276      IF (NEND.EQ.1) THEN
277          CALL=OPLOCAT(11+1)
278      ELSE
279          CALL=OPLOCAT(11+1)
280      ENDIF
281
282      GO TO 1
283
284      CALL=OPLOCAT(11+1)
285
286      IF (NEND.EQ.1) THEN
287          CALL=OPLOCAT(11+1)
288      ELSE
289          CALL=OPLOCAT(11+1)
290      ENDIF
291
292      GO TO 1
293
294      CALL=OPLOCAT(11+1)
295
296      IF (NEND.EQ.1) THEN
297          CALL=OPLOCAT(11+1)
298      ELSE
299          CALL=OPLOCAT(11+1)
300      ENDIF
301
302      GO TO 1
303
304      CALL=OPLOCAT(11+1)
305
306      IF (NEND.EQ.1) THEN
307          CALL=OPLOCAT(11+1)
308      ELSE
309          CALL=OPLOCAT(11+1)
310      ENDIF
311
312      GO TO 1
313
314      CALL=OPLOCAT(11+1)
315
316      IF (NEND.EQ.1) THEN
317          CALL=OPLOCAT(11+1)
318      ELSE
319          CALL=OPLOCAT(11+1)
320      ENDIF
321
322      GO TO 1
323
324      CALL=OPLOCAT(11+1)
325
326      IF (NEND.EQ.1) THEN
327          CALL=OPLOCAT(11+1)
328      ELSE
329          CALL=OPLOCAT(11+1)
330      ENDIF
331
332      GO TO 1
333
334      CALL=OPLOCAT(11+1)
335
336      IF (NEND.EQ.1) THEN
337          CALL=OPLOCAT(11+1)
338      ELSE
339          CALL=OPLOCAT(11+1)
340      ENDIF
341
342      GO TO 1
343
344      CALL=OPLOCAT(11+1)
345
346      IF (NEND.EQ.1) THEN
347          CALL=OPLOCAT(11+1)
348      ELSE
349          CALL=OPLOCAT(11+1)
350      ENDIF
351
352      GO TO 1
353
354      CALL=OPLOCAT(11+1)
355
356      IF (NEND.EQ.1) THEN
357          CALL=OPLOCAT(11+1)
358      ELSE
359          CALL=OPLOCAT(11+1)
360      ENDIF
361
362      GO TO 1
363
364      CALL=OPLOCAT(11+1)
365
366      IF (NEND.EQ.1) THEN
367          CALL=OPLOCAT(11+1)
368      ELSE
369          CALL=OPLOCAT(11+1)
370      ENDIF
371
372      GO TO 1
373
374      CALL=OPLOCAT(11+1)
375
376      IF (NEND.EQ.1) THEN
377          CALL=OPLOCAT(11+1)
378      ELSE
379          CALL=OPLOCAT(11+1)
380      ENDIF
381
382      GO TO 1
383
384      CALL=OPLOCAT(11+1)
385
386      IF (NEND.EQ.1) THEN
387          CALL=OPLOCAT(11+1)
388      ELSE
389          CALL=OPLOCAT(11+1)
390      ENDIF
391
392      GO TO 1
393
394      CALL=OPLOCAT(11+1)
395
396      IF (NEND.EQ.1) THEN
397          CALL=OPLOCAT(11+1)
398      ELSE
399          CALL=OPLOCAT(11+1)
400      ENDIF
401
402      GO TO 1
403
404      CALL=OPLOCAT(11+1)
405
406      IF (NEND.EQ.1) THEN
407          CALL=OPLOCAT(11+1)
408      ELSE
409          CALL=OPLOCAT(11+1)
410      ENDIF
411
412      GO TO 1
413
414      CALL=OPLOCAT(11+1)
415
416      IF (NEND.EQ.1) THEN
417          CALL=OPLOCAT(11+1)
418      ELSE
419          CALL=OPLOCAT(11+1)
420      ENDIF
421
422
```

Listing of LORAN FOR at 02-10-23 on APR 12, 1966 for CE-ROCKHAW ON DELTANTS

Continued on reverse

```

733 C .....
734 DO 300 J=J0+1, J0+1
735 J0=J+1
736 DO 300 I=I0+1, I0+1
737 I0=I+1
738 C
739 I0=I0+1, J0=J0+1
740 J0=J0+1, I0=I0+1
741 J0=J0+1
742 J0=J0+1
743 C
744 IF ((I, LV, I0) OR (I, LV, I0) OR
745     (I, LV, J0) OR (I, LV, J0) THEN
746 C .....
747 OUT OF TOTAL REGION: G10, G1/G10+1, G1
748 C .....
749 CALL F000(I, J, I0, J0)
750 GO TO 300
751 ENDIF
752 C
753 IF ((I, LV, I0) AND (I, LV, I0) AND (I, LV, J0)
754     AND (I, LV, J0) THEN
755 C .....
756 INNER POINTS IN THE TOTAL REGION: G10+1, G1
757 C .....
758 DO 310 I=I0+1, I0+1
759 C .....
760 C .....
761 C .....
762 C .....
763 C .....
764 C .....
765 C .....
766 C .....
767 C .....
768 C .....
769 C .....
770 C .....
771 CALL F000(I, J, I0, J0, F000, F000, F000)
772 PP10, I0=PP10, I0=PP10
773 GO TO 300
774 ENDIF
775 C
776 IF ((I, LV, I0) AND (I, LV, J0) AND (I, LV, J0) THEN
777 C .....
778 ON LEFT BOUNDARY OF THE TOTAL REGION
779 C .....
780 CALL F000(I, J, I0, J0, F000, F000, F000)
781 DO 320 I=I0+1, I0+1
782 C .....
783 PP10, I0=PP10, I0=PP10
784 C .....
785 C .....
786 C .....
787 C .....
788 C .....
789 CALL F000(I, J, I0, J0, F000, F000, F000)
790 PP10, I0=PP10, I0=PP10
791 GO TO 300
792 ENDIF
793 C
794 IF ((I, LV, I0) AND (I, LV, J0) AND (I, LV, J0) THEN
795 C .....
796 ON LOWER BOUNDARY OF THE TOTAL REGION
797 C .....
798 CALL F000(I, J, I0, J0, F000, F000, F000)
799 DO 330 I=I0+1, I0+1
800 C .....
801 PP10, I0=PP10, I0=PP10
802 C .....
803 C .....
804 C .....
805 C .....
806 C .....
807 CALL F000(I, J, I0, J0, F000, F000, F000)
808 PP10, I0=PP10, I0=PP10
809 GO TO 300
810 ENDIF
811 C
812 IF ((I, LV, I0) AND (I, LV, J0) THEN
813 C .....
814 ON LOWER LEFT CORNER OF THE TOTAL REGION
815 C .....
816 CALL F000(I, J, I0, J0, F000, F000, F000)
817 DO 340 I=I0+1, I0+1
818 C .....
819 IF (I0=I0) THEN
820 C .....
821 PP10, I0=PP10, I0=PP10
822 C .....
823 C .....
824 C .....
825 C .....
826 C .....
827 CALL F000(I, J, I0, J0, F000, F000, F000)
828 PP10, I0=PP10, I0=PP10
829 GO TO 300
830 ENDIF
831 IF (I0=I0) THEN
832 C .....
833 PP10, I0=PP10, I0=PP10
834 C .....
835 C .....
836 C .....
837 C .....
838 C .....
839 CALL F000(I, J, I0, J0, F000, F000, F000)
840 PP10, I0=PP10, I0=PP10
841 GO TO 300
842 ENDIF
843 IF (I0=I0) THEN
844 C .....
845 PP10, I0=PP10, I0=PP10
846 C .....
847 C .....
848 C .....
849 C .....
850 C .....
851 CALL F000(I, J, I0, J0, F000, F000, F000)
852 PP10, I0=PP10, I0=PP10
853 GO TO 300
854 ENDIF
855 C
856 C .....
857 C .....
858 C .....
859 C .....
860 C .....
861 C .....
862 C .....
863 C .....
864 C .....
865 C .....
866 C .....
867 C .....
868 C .....
869 C .....
870 C .....
871 C .....
872 C .....
873 C .....
874 C .....
875 C .....
876 C .....
877 C .....
878 C .....
879 C .....
880 C .....
881 C .....
882 C .....
883 C .....
884 C .....
885 C .....
886 C .....
887 C .....
888 C .....
889 C .....
890 C .....
891 C .....
892 C .....
893 C .....
894 C .....
895 C .....
896 C .....
897 C .....
898 C .....
899 C .....
900 C .....
901 C .....
902 C .....
903 C .....
904 C .....
905 C .....
906 C .....
907 C .....
908 C .....
909 C .....
910 C .....
911 C .....
912 C .....
913 C .....
914 C .....
915 C .....
916 C .....
917 C .....
918 C .....
919 C .....
920 C .....
921 C .....
922 C .....
923 C .....
924 C .....
925 C .....
926 C .....
927 C .....
928 C .....
929 C .....
930 C .....
931 C .....
932 C .....
933 C .....
934 C .....
935 C .....
936 C .....
937 C .....
938 C .....
939 C .....
940 C .....
941 C .....
942 C .....
943 C .....
944 C .....
945 C .....
946 C .....
947 C .....
948 C .....
949 C .....
950 C .....
951 C .....
952 C .....
953 C .....
954 C .....
955 C .....
956 C .....
957 C .....
958 C .....
959 C .....
960 C .....
961 C .....
962 C .....
963 C .....
964 C .....
965 C .....
966 C .....
967 C .....
968 C .....
969 C .....
970 C .....
971 C .....
972 C .....
973 C .....
974 C .....
975 C .....
976 C .....
977 C .....
978 C .....
979 C .....
980 C .....
981 C .....
982 C .....
983 C .....
984 C .....
985 C .....
986 C .....
987 C .....
988 C .....
989 C .....
990 C .....
991 C .....
992 C .....
993 C .....
994 C .....
995 C .....
996 C .....
997 C .....
998 C .....
999 C .....
1000 C .....

```


Warrant in Arrest

[illegible]

[illegible]

University of Alberta

```

1000 C
1001 C
1002 C
1003 C
1004 C
1005 C
1006 C
1007 C
1008 C
1009 C
1010 C
1011 C
1012 C
1013 C
1014 C
1015 C
1016 C
1017 C
1018 C
1019 C
1020 C
1021 C
1022 C
1023 C
1024 C
1025 C
1026 C
1027 C
1028 C
1029 C
1030 C
1031 C
1032 C
1033 C
1034 C
1035 C
1036 C
1037 C
1038 C
1039 C
1040 C
1041 C
1042 C
1043 C
1044 C
1045 C
1046 C
1047 C
1048 C
1049 C
1050 C
1051 C
1052 C
1053 C
1054 C
1055 C
1056 C
1057 C
1058 C
1059 C
1060 C
1061 C
1062 C
1063 C
1064 C
1065 C
1066 C
1067 C
1068 C
1069 C
1070 C
1071 C
1072 C
1073 C
1074 C
1075 C
1076 C
1077 C
1078 C
1079 C
1080 C
1081 C
1082 C
1083 C
1084 C
1085 C
1086 C
1087 C
1088 C
1089 C
1090 C
1091 C
1092 C
1093 C
1094 C
1095 C
1096 C
1097 C
1098 C
1099 C
1100 C
1101 C
1102 C
1103 C
1104 C
1105 C
1106 C
1107 C
1108 C
1109 C
1110 C
1111 C
1112 C
1113 C
1114 C
1115 C
1116 C
1117 C
1118 C
1119 C
1120 C
1121 C
1122 C
1123 C
1124 C
1125 C
1126 C
1127 C
1128 C
1129 C
1130 C
1131 C
1132 C
1133 C
1134 C
1135 C
1136 C
1137 C
1138 C
1139 C
1140 C
1141 C
1142 C
1143 C
1144 C
1145 C
1146 C
1147 C
1148 C
1149 C
1150 C
1151 C
1152 C
1153 C
1154 C
1155 C
1156 C
1157 C
1158 C
1159 C
1160 C
1161 C
1162 C
1163 C
1164 C
1165 C
1166 C
1167 C
1168 C
1169 C
1170 C
1171 C
1172 C
1173 C
1174 C
1175 C
1176 C
1177 C
1178 C
1179 C
1180 C
1181 C
1182 C
1183 C
1184 C
1185 C
1186 C
1187 C
1188 C
1189 C
1190 C
1191 C
1192 C
1193 C
1194 C
1195 C
1196 C
1197 C
1198 C
1199 C
1200 C
1201 C
1202 C
1203 C
1204 C
1205 C
1206 C
1207 C
1208 C
1209 C
1210 C
1211 C
1212 C
1213 C
1214 C
1215 C
1216 C
1217 C
1218 C
1219 C
1220 C
1221 C
1222 C
1223 C
1224 C
1225 C
1226 C
1227 C
1228 C
1229 C
1230 C
1231 C
1232 C
1233 C
1234 C
1235 C
1236 C
1237 C
1238 C
1239 C
1240 C
1241 C
1242 C
1243 C
1244 C
1245 C
1246 C
1247 C
1248 C
1249 C
1250 C
1251 C
1252 C
1253 C
1254 C
1255 C
1256 C
1257 C
1258 C
1259 C
1260 C
1261 C
1262 C
1263 C
1264 C
1265 C
1266 C
1267 C
1268 C
1269 C
1270 C
1271 C
1272 C
1273 C
1274 C
1275 C
1276 C
1277 C
1278 C
1279 C
1280 C
1281 C
1282 C
1283 C
1284 C
1285 C
1286 C
1287 C
1288 C
1289 C
1290 C
1291 C
1292 C
1293 C
1294 C
1295 C
1296 C
1297 C
1298 C
1299 C
1300 C
1301 C
1302 C
1303 C
1304 C
1305 C
1306 C
1307 C
1308 C
1309 C
1310 C
1311 C
1312 C
1313 C
1314 C
1315 C
1316 C
1317 C
1318 C
1319 C
1320 C
1321 C
1322 C
1323 C
1324 C
1325 C
1326 C
1327 C
1328 C
1329 C
1330 C
1331 C
1332 C
1333 C
1334 C
1335 C
1336 C
1337 C
1338 C
1339 C
1340 C
1341 C
1342 C
1343 C
1344 C
1345 C
1346 C
1347 C
1348 C
1349 C
1350 C
1351 C
1352 C
1353 C
1354 C
1355 C
1356 C
1357 C
1358 C
1359 C
1360 C
1361 C
1362 C
1363 C
1364 C
1365 C
1366 C
1367 C
1368 C
1369 C
1370 C
1371 C
1372 C
1373 C
1374 C
1375 C
1376 C
1377 C
1378 C
1379 C
1380 C
1381 C
1382 C
1383 C
1384 C
1385 C
1386 C
1387 C
1388 C
1389 C
1390 C
1391 C
1392 C
1393 C
1394 C
1395 C
1396 C
1397 C
1398 C
1399 C
1400 C
1401 C
1402 C
1403 C
1404 C
1405 C
1406 C
1407 C
1408 C
1409 C
1410 C
1411 C
1412 C
1413 C
1414 C
1415 C
1416 C
1417 C
1418 C
1419 C
1420 C
1421 C
1422 C
1423 C
1424 C
1425 C
1426 C
1427 C
1428 C
1429 C
1430 C
1431 C
1432 C
1433 C
1434 C
1435 C
1436 C
1437 C
1438 C
1439 C
1440 C
1441 C
1442 C
1443 C
1444 C
1445 C
1446 C
1447 C
1448 C
1449 C
1450 C
1451 C
1452 C
1453 C
1454 C
1455 C
1456 C
1457 C
1458 C
1459 C
1460 C
1461 C
1462 C
1463 C
1464 C
1465 C
1466 C
1467 C
1468 C
1469 C
1470 C
1471 C
1472 C
1473 C
1474 C
1475 C
1476 C
1477 C
1478 C
1479 C
1480 C
1481 C
1482 C
1483 C
1484 C
1485 C
1486 C
1487 C
1488 C
1489 C
1490 C
1491 C
1492 C
1493 C
1494 C
1495 C
1496 C
1497 C
1498 C
1499 C
1500 C
1501 C
1502 C
1503 C
1504 C
1505 C
1506 C
1507 C
1508 C
1509 C
1510 C
1511 C
1512 C
1513 C
1514 C
1515 C
1516 C
1517 C
1518 C
1519 C
1520 C
1521 C
1522 C
1523 C
1524 C
1525 C
1526 C
1527 C
1528 C
1529 C
1530 C
1531 C
1532 C
1533 C
1534 C
1535 C
1536 C
1537 C
1538 C
1539 C
1540 C
1541 C
1542 C
1543 C
1544 C
1545 C
1546 C
1547 C
1548 C
1549 C
1550 C
1551 C
1552 C
1553 C
1554 C
1555 C
1556 C
1557 C
1558 C
1559 C
1560 C
1561 C
1562 C
1563 C
1564 C
1565 C
1566 C
1567 C
1568 C
1569 C
1570 C
1571 C
1572 C
1573 C
1574 C
1575 C
1576 C
1577 C
1578 C
1579 C
1580 C
1581 C
1582 C
1583 C
1584 C
1585 C
1586 C
1587 C
1588 C
1589 C
1590 C
1591 C
1592 C
1593 C
1594 C
1595 C
1596 C
1597 C
1598 C
1599 C
1600 C
1601 C
1602 C
1603 C
1604 C
1605 C
1606 C
1607 C
1608 C
1609 C
1610 C
1611 C
1612 C
1613 C
1614 C
1615 C
1616 C
1617 C
1618 C
1619 C
1620 C
1621 C
1622 C
1623 C
1624 C
1625 C
1626 C
1627 C
1628 C
1629 C
1630 C
1631 C
1632 C
1633 C
1634 C
1635 C
1636 C
1637 C
1638 C
1639 C
1640 C
1641 C
1642 C
1643 C
1644 C
1645 C
1646 C
1647 C
1648 C
1649 C
1650 C
1651 C
1652 C
1653 C
1654 C
1655 C
1656 C
1657 C
1658 C
1659 C
1660 C
1661 C
1662 C
1663 C
1664 C
1665 C
1666 C
1667 C
1668 C
1669 C
1670 C
1671 C
1672 C
1673 C
1674 C
1675 C
1676 C
1677 C
1678 C
1679 C
1680 C
1681 C
1682 C
1683 C
1684 C
1685 C
1686 C
1687 C
1688 C
1689 C
1690 C
1691 C
1692 C
1693 C
1694 C
1695 C
1696 C
1697 C
1698 C
1699 C
1700 C
1701 C
1702 C
1703 C
1704 C
1705 C
1706 C
1707 C
1708 C
1709 C
1710 C
1711 C
1712 C
1713 C
1714 C
1715 C
1716 C
1717 C
1718 C
1719 C
1720 C
1721 C
1722 C
1723 C
1724 C
1725 C
1726 C
1727 C
1728 C
1729 C
1730 C
1731 C
1732 C
1733 C
1734 C
1735 C
1736 C
1737 C
1738 C
1739 C
1740 C
1741 C
1742 C
1743 C
1744 C
1745 C
1746 C
1747 C
1748 C
1749 C
1750 C
1751 C
1752 C
1753 C
1754 C
1755 C
1756 C
1757 C
1758 C
1759 C
1760 C
1761 C
1762 C
1763 C
1764 C
1765 C
1766 C
1767 C
1768 C
1769 C
1770 C
1771 C
1772 C
1773 C
1774 C
1775 C
1776 C
1777 C
1778 C
1779 C
1780 C
1781 C
1782 C
1783 C
1784 C
1785 C
1786 C
1787 C
1788 C
1789 C
1790 C
1791 C
1792 C
1793 C
1794 C
1795 C
1796 C
1797 C
1798 C
1799 C
1800 C
1801 C
1802 C
1803 C
1804 C
1805 C
1806 C
1807 C
1808 C
1809 C
1810 C
1811 C
1812 C
1813 C
1814 C
1815 C
1816 C
1817 C
1818 C
1819 C
1820 C
1821 C
1822 C
1823 C
1824 C
1825 C
1826 C
1827 C
1828 C
1829 C
1830 C
1831 C
1832 C
1833 C
1834 C
1835 C
1836 C
1837 C
1838 C
1839 C
1840 C
1841 C
1842 C
1843 C
1844 C
1845 C
1846 C
1847 C
1848 C
1849 C
1850 C
1851 C
1852 C
1853 C
1854 C
1855 C
1856 C
1857 C
1858 C
1859 C
1860 C
1861 C
1862 C
1863 C
1864 C
1865 C
1866 C
1867 C
1868 C
1869 C
1870 C
1871 C
1872 C
1873 C
1874 C
1875 C
1876 C
1877 C
1878 C
1879 C
1880 C
1881 C
1882 C
1883 C
1884 C
1885 C
1886 C
1887 C
1888 C
1889 C
1890 C
1891 C
1892 C
1893 C
1894 C
1895 C
1896 C
1897 C
1898 C
1899 C
1900 C
1901 C
1902 C
1903 C
1904 C
1905 C
1906 C
1907 C
1908 C
1909 C
1910 C
1911 C
1912 C
1913 C
1914 C
1915 C
1916 C
1917 C
1918 C
1919 C
1920 C
1921 C
1922 C
1923 C
1924 C
1925 C
1926 C
1927 C
1928 C
1929 C
1930 C
1931 C
1932 C
1933 C
1934 C
1935 C
1936 C
1937 C
1938 C
1939 C
1940 C
1941 C
1942 C
1943 C
1944 C
1945 C
1946 C
1947 C
1948 C
1949 C
1950 C
1951 C
1952 C
1953 C
1954 C
1955 C
1956 C
1957 C
1958 C
1959 C
1960 C
1961 C
1962 C
1963 C
1964 C
1965 C
1966 C
1967 C
1968 C
1969 C
1970 C
1971 C
1972 C
1973 C
1974 C
1975 C
1976 C
1977 C
1978 C
1979 C
1980 C
1981 C
1982 C
1983 C
1984 C
1985 C
1986 C
1987 C
1988 C
1989 C
1990 C
1991 C
1992 C
1993 C
1994 C
1995 C
1996 C
1997 C
1998 C
1999 C
2000 C

```

[illegible]

Listing of LAMB.POB at 00:10:23 on APR 12, 1990 for CE+PHEEN on 00178775

```

1343 C
1344 WRITE(0,000) 1.0,0.000000,1.000000,0.010,1.000000,0.000000,1.000000
1345 10 CONTINUE
1346 100 CONTINUE
1347 C
1348 WRITE(0,000) 0.000
1349 C
1350 000 FORMAT(1X,'RESULTS ON LEVEL ',12,' IS:')
1351 010 FORMAT(1X,'00','00','00','00','00','00','00','00','00','00','00','00')
1352 020 FORMAT(1X,'0120.101.20.01010.0.0000')
1353 030 FORMAT(1X,'THE MAXIMUM ERROR ON LEVEL ',12,' IS ',010.0)
1354 C
1355 RETURN
1356 END

```