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.

S'il 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, 1990

ISBN 0-315-64969-8

UNIVERSITY OF ALBERTA

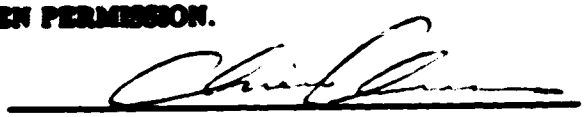
RELEASE FORM

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



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

1. Introduction.

Consider the elliptic differential equation:

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

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:

$$L_h u_h = f_h, \qquad x \in \Omega_h \subset \Omega \tag{1.2}$$

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

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

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:

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

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:

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

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

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

and

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

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[4,5,7] and McCormick[8] 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:

$$e_h := u_h^* - v_h \tag{2.1}$$

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:

$$r_h := f_h - L_h v_h \tag{2.2}$$

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:

$$L_h e_h = r_h \tag{2.3}$$

which provides us with the very important fact that the error e_h satisfies the same set of equations as the unknown u_h, when f_h is replaced by the residual r_h. 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_h, we may improve the approximation v_h by a new approximation $\bar{v}_h$ defined by:

$$\bar{v}_h := v_h + e_h \tag{2.4}$$

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_h denote the diagonal part of L_h, i.e.

$$D_h := diag(L_h)$$

The damped Jacobi iteration method for (1.2) is:

$$v_h^{m+1} := v_h^m + \omega D_h^{-1}(f_h - L_h v_h^m) \tag{2.5}$$

where, v_h^m is the mth approximation of v_h to u_h^* on grid Ω_h, and $\omega \in [0,1]$ is a damping parameter.

For exact solution of (1.2), we have:

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

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

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

where,

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

Denote the iteration matrix as S_h, i.e.

$$S_h := I - \omega D_h^{-1} L_h \tag{2.8}$$

we have:

$$\begin{aligned} e_h^{m+1} &= S_h e_h^m \\ &= S_h^2 e_h^{m-1} \\ &\cdots \\ &= S_h^{m+1} e_h^0 \end{aligned} \tag{2.9}$$

Therefore,

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

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

$$\rho(S_h) := \max_i |\lambda_i(S_h)| \tag{2.10}$$

where $\lambda_i(S_h)$ is the ith eigenvalue of S_h, the following is true:

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

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

The term $\rho(S_h)$ 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 $\rho(S_h) < 1$ and let M be the smallest integer satisfying

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

by solving for M, we have:

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

Therefore, the convergence factor $\rho(S_h)$ could be used to calculate the number of iterations needed to reduce the error by a factor of 10^{-t}. The term $-log_{10}\,[\rho(S_h)]$ on the denominator in (2.11) is called the covergence 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 $\rho(S_h)$, 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_h defined by (1.4) are

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

$$w_{pq}(L_h) = \sin(p\pi x_h)\sin(q\pi y_h), \quad (x_h, y_h) \in \Omega_h \tag{2.13}$$

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:

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

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

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

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

$$e_h^0 := \sum_{1 \le p,q \le n-1} a_{pq} w_{pq}(L_h) \tag{2.15}$$

where a_{pq} are the proper weighting factors.

On the right hand side of (2.15), the components $a_{pq}w_{pq}(L_h)$ corresponding to $1 \le p, q \le \frac{n}{2} - 1$ are called low frequency or smooth error components of e_h^0 while those corresponding to $\frac{n}{2} \le p, q \le n-1$ 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:

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

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

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

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

which approaches one when $h \to 0$ for any $0 < \omega \le 1$. 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 $0 < \omega \le 1$. 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 $\lambda_{n-1,n-1}(S_h)$:

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

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

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

Denote

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

we have

$$L_h e_h = r_h \tag{2.18}$$

where,

$$r_h := f_h - L_h \bar{v}_h \tag{2.19}$$

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

$$u_h^* = v_h + e_h^*$$

Unfortunately, to solve (2.18) exactly is as difficult as to solve (1.2). Nevertheless, by the previous discussion, we know that e_h is a smooth function, and so is r_h. 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_h to u_h^*,

(1) Calculation of residual:

$$r_h := f_h - L_h v_h;$$

(2) Restriction of residual:

$$r_H := R_h^H r_h;$$

(3) Exact solution of coarse-grid equation:

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

(4) Prolongation of correction:

$$e_h := I_H^h e_H;$$

(5) Correction of v_h:

$$\bar{v}_h := v_h + e_h$$

where,

$R_h^H : G(\Omega_h) \to G(\Omega_H)$ is the restriction operator;

$I_H^h : G(\Omega_H) \to G(\Omega_h)$ is the interpolation operator; and

L_H is the discretized 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) $r_h := f_h - L_h \bar{v}_h$;

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

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

(2.d) $e_h := I_H^h e_H$;

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

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,

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

and for two-dimensional problems, we have:

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

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

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

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

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

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:

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

and for two-dimensional problems, we have

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

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_{H_0}. Then, on that coarsest grid, it should be quite cheap to solve for e_{H_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.[10],[11],[12] Nevertheless, instead of solving nonlinear problems directly, we have to spend considerable effort in carrying out the Newton

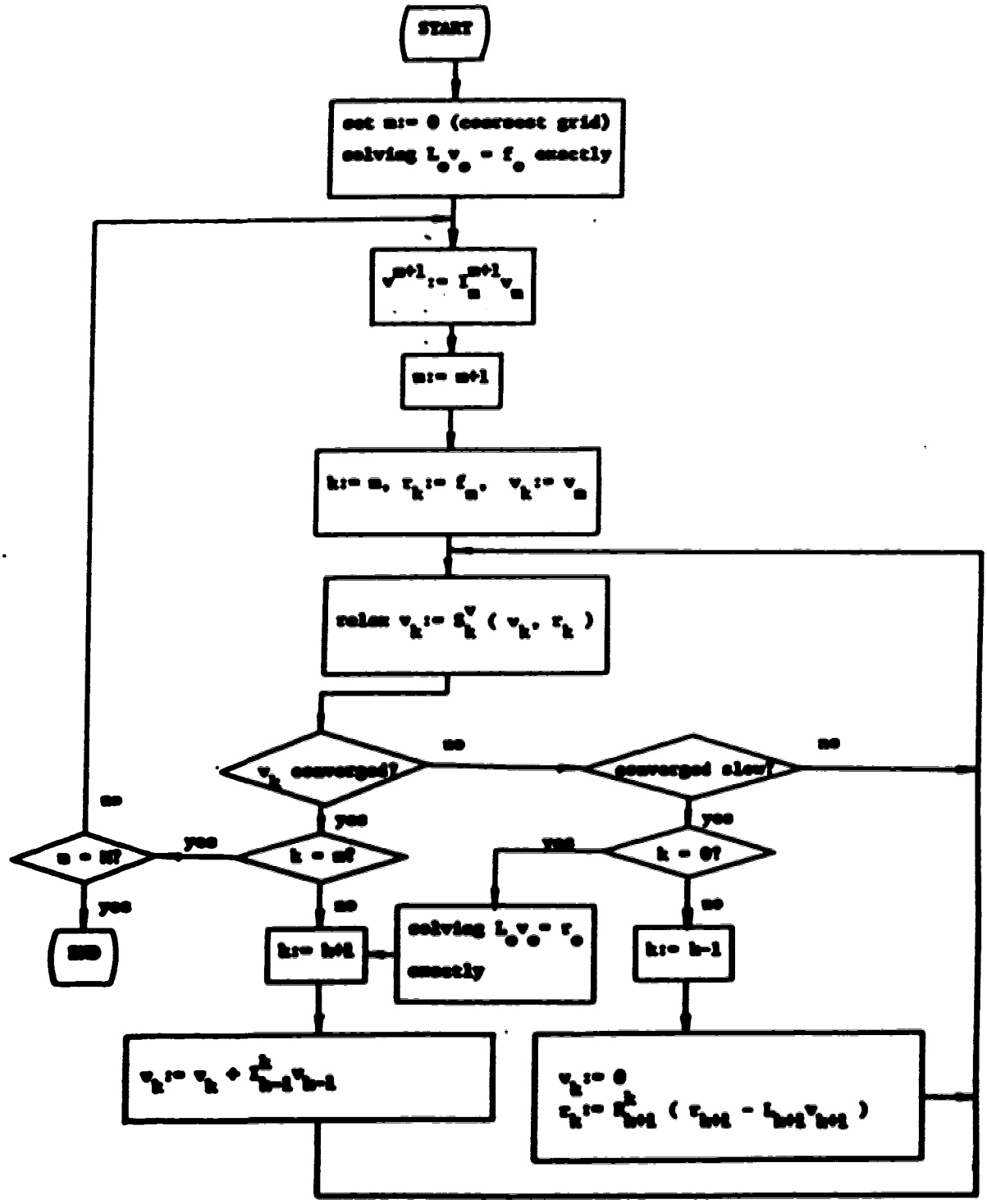


Figure 2.1

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:

$$L_h u_h = f_h, \qquad x \in \Omega_h \subset \Omega \tag{2.23}$$

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

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

is a nonlinear operator.

The residual equation is as follows:

$$\hat{L}_h e_h = r_h \tag{2.24}$$

where,

$$\hat{L}_h e_h \equiv L_h(v_h + e_h) - L_h v_h \tag{2.25}$$

and

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

By (2.25), we know that $\hat{L}_h = L_h$ 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:

$$L_H(I_h^H v_h + e_H) - L_H(I_h^H v_h) = I_h^H r_h \tag{2.26}$$

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

Let

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

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

by (2.26), we have

$$L_H v_H = f_h^H \tag{2.27}$$

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

$$v_h := v_h + I_H^h(v_H - R_h^H v_h) \tag{2.28}$$

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

$$f_h^H := f_H + \tau_h^H \tag{2.29}$$

where,

$$\tau_h^H := L_H(R_h^H v_h) - R_h^H(L_h v_h) \tag{2.30}$$

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 technique.

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

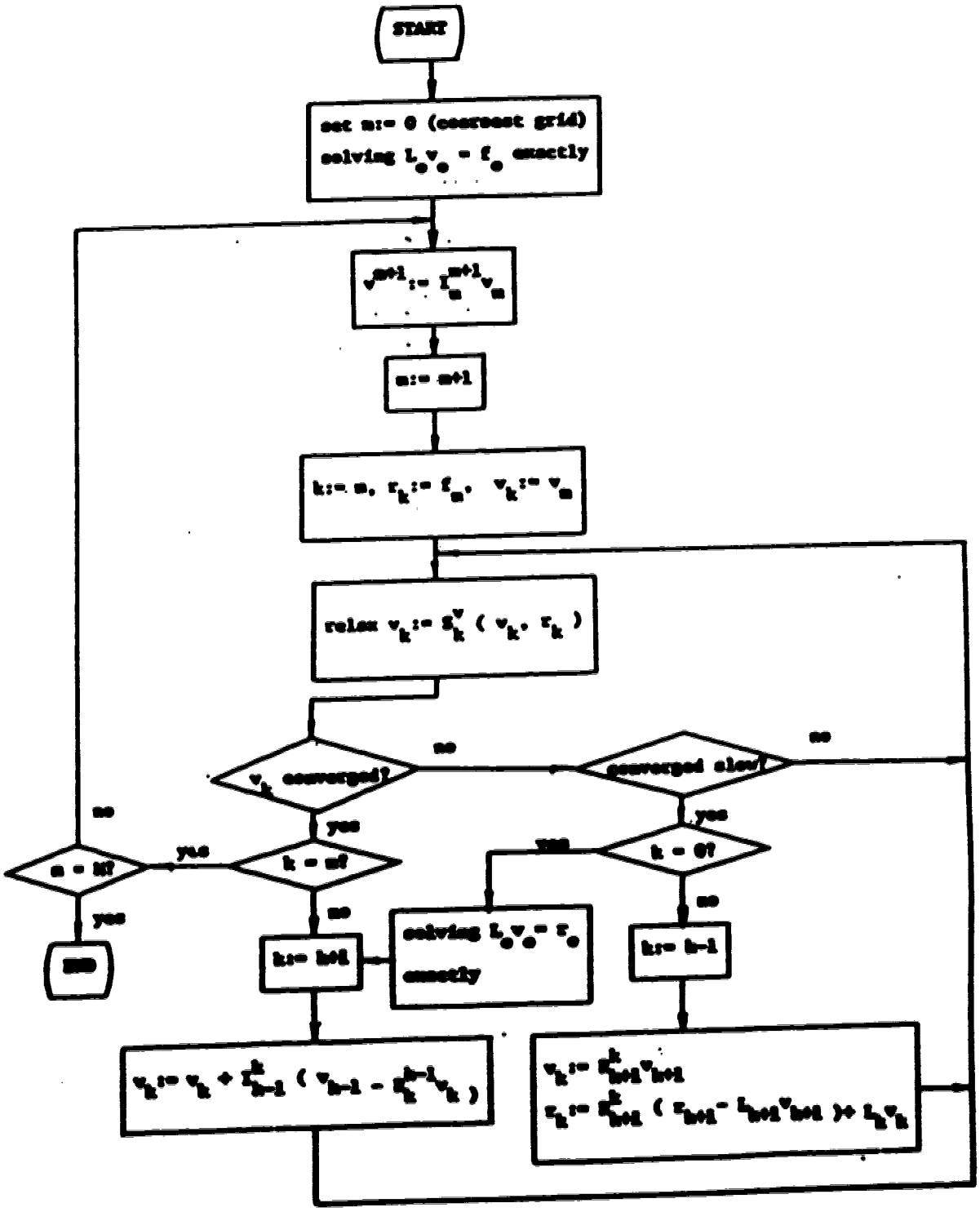


Figure 2.2

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)

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

with

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

and

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

where α is a parameter.

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

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

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.[9] 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;[2],[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.

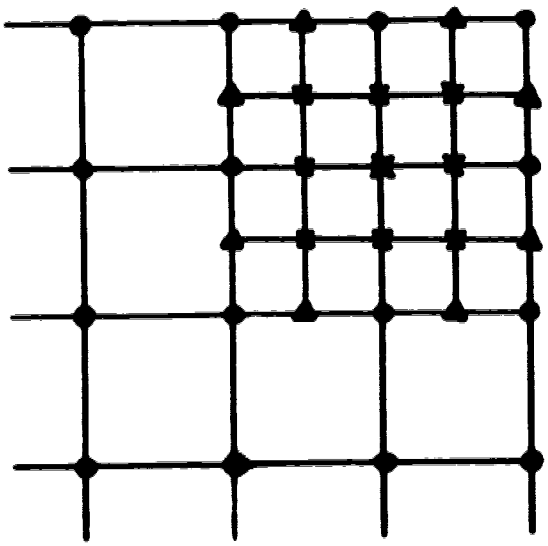
3.2 Composite grid structure.

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, $\Omega_0, \Omega_1, \ldots, \Omega_M$, 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:

$$\Omega_{comp} = \Omega_0 \cup \Omega_1 \cup \ldots \cup \Omega_k \cup \Omega_{k+1} \cup \ldots \cup \Omega_M \tag{3.3}$$

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



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

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_{comp}, 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.

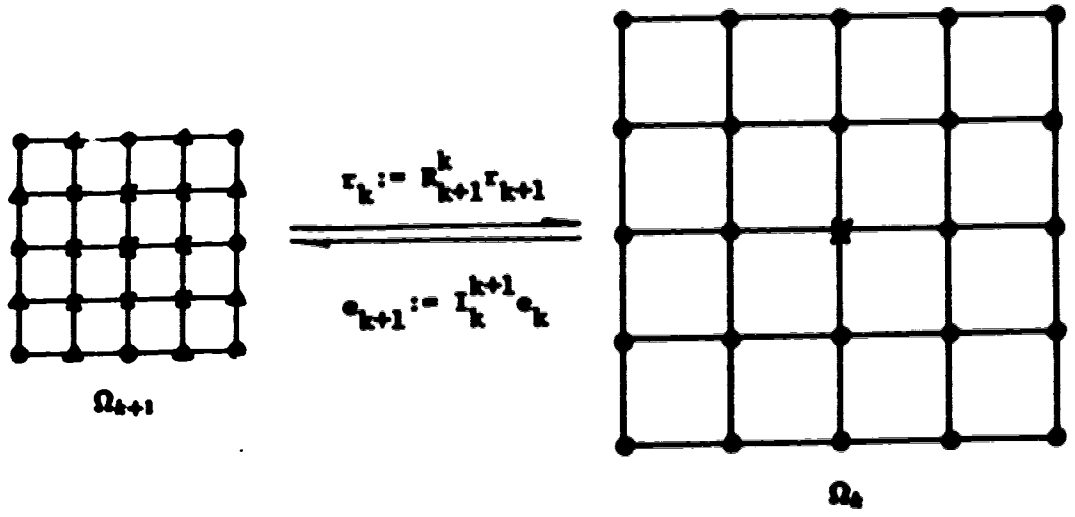


Figure 3.2

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

$$L_{k+1} u_{k+1} = f_{k+1}, \qquad x \in \Omega_{k+1} \subset \Omega \tag{3.4}$$

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

(1) Pre-relaxation (ν_1 times):

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

(2) Coarse-grid correction:

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

(3) Correction:

$$v_{k+1} := v_{k+1} + I_k^{k+1}(v_k - R_{k+1}^k v_{k+1}) \tag{3.7}$$

(4) Post-relaxation (ν_2 times):

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

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

$$\tau_k^{k+1} := L_k(R_{k+1}^k v_{k+1}) - R_{k+1}^k(L_{k+1} v_{k+1}) \tag{3.9}$$

It is clear that the term τ_k^{k+1} is the truncation error on Ω_k relative to Ω_{k+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_{k+1} to u_{k+1}° on the local region grid Ω_{k+1},

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

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

(2) Coarse-grid correction:

(a) Calculate residuals on the inner points of local region Ω_{k+1} (corresponding to points × and ⊗):

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

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

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

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

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

$$r_k := R_{k+1}^k r_{k+1};$$

(d) Solve v_k on course grid Ω_k:

$$v_k := L_k^{-1}(L_k(R_{k+1}^k v_{k+1}) + r_k);$$

(3) Correction:

$$v_{k+1} := v_{k+1} + I_k^{k+1}(v_k - R_{k+1}^k v_{k+1});$$

(4) Post-relaxation (ν_2 times):

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

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.[3,4,5,7]

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

Finally, in the course-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_k as we do on the course grid Ω_k. 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

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

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

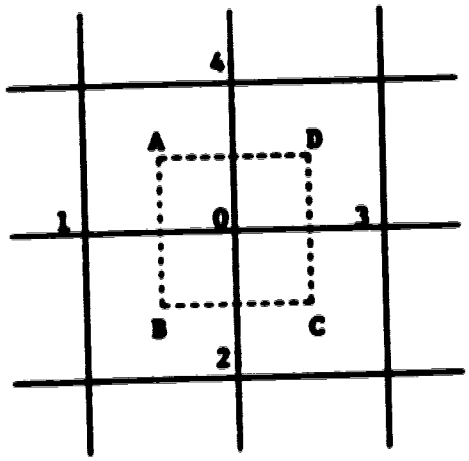


Figure 3.3

Finite volume method for inner points

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

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

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

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

where,

$$f_{k+1}^{(0)} = \iint_{\Omega_0} f \, dx \, dy.$$

(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*.

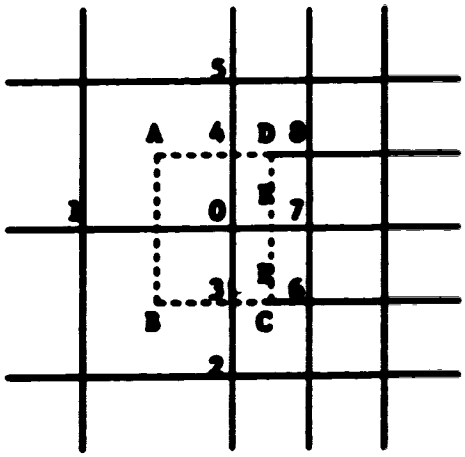


Figure 3.4

Seven-point finite volume method

for inner boundary (but not corner) points

From *Figure 3.4*, we have:

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

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

$$\int_{CD} \frac{\partial u}{\partial n}\,dl \approx \frac{v_{k+1}^{(6)} - v_{k+1}^{(2)}}{h_{k+1}} \cdot \frac{1}{2}h_{k+1} + \frac{v_{k+1}^{(7)} - v_{k+1}^{(0)}}{h_{k+1}} \cdot h_{k+1} + \frac{v_{k+1}^{(8)} - v_{k+1}^{(4)}}{h_{k+1}} \cdot \frac{1}{2}h_{k+1} \tag{3.13}$$

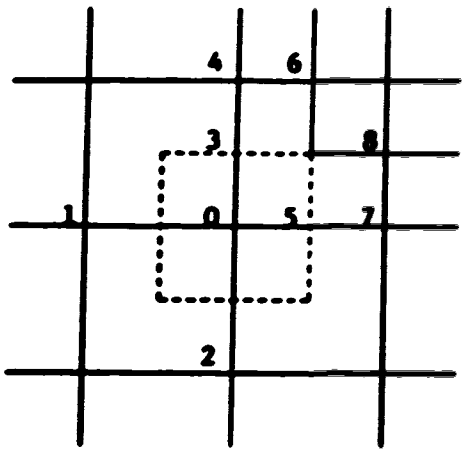
This yields the following approximation $L_{k+1}(v_{k+1}, v_k)$ to Lu on the inner boundary points (excluding the corners):

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

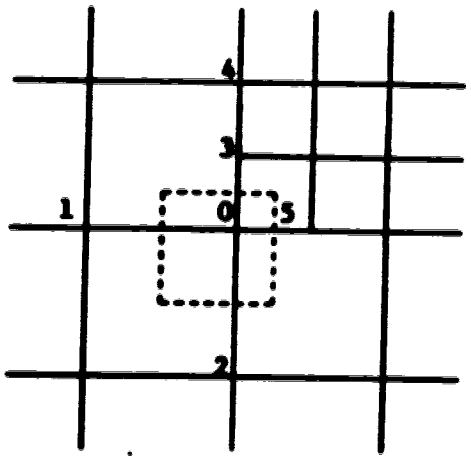
(Here, we have used $h_k = 2h_{k+1}$ for the simplification.)

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,

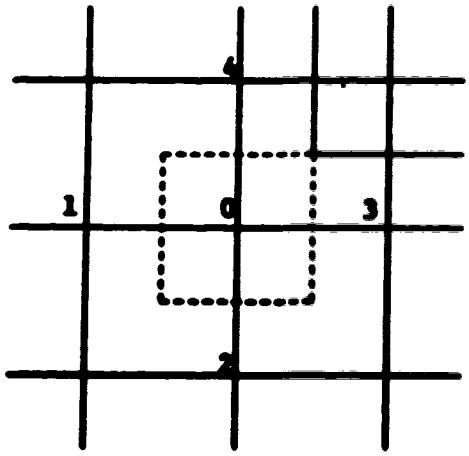
$$\begin{aligned} L_{k+1}(v_{k+1}, v_k) \approx\ & \frac{v_k^{(1)} - v_{k+1}^{(0)}}{h_k} \cdot h_k + \frac{v_k^{(2)} - v_{k+1}^{(0)}}{h_k} \cdot h_k \\ & + \frac{v_{k+1}^{(7)} - v_{k+1}^{(0)}}{2h_{k+1}} \cdot (\frac{h_k}{2} + \frac{h_{k+1}}{2}) + \frac{v_{k+1}^{(8)} - v_{k+1}^{(2)}}{2h_{k+1}} \cdot \frac{h_{k+1}}{2} \\ & + \frac{v_{k+1}^{(0)} - v_{k+1}^{(5)}}{2h_{k+1}} \cdot \frac{h_{k+1}}{2} + \frac{\blacksquare - \blacksquare}{\blacksquare} \cdot (\frac{h_k}{2} + \frac{h_{k+1}}{2}) \\ =\ & (v_k^{(1)} + v_k^{(2)} - 2v_{k+1}^{(0)}) + \frac{3}{4}(v_{k+1}^{(6)} + v_{k+1}^{(7)} - 2v_{k+1}^{(0)}) \\ & + \frac{1}{4}(v_{k+1}^{(8)} + v_{k+1}^{(0)} - v_{k+1}^{(2)} - v_{k+1}^{(5)}) \end{aligned} \tag{3.15}$$



(a)



(b)



(c)

Figure 3.5

Finite volume method

for inner boundary corner points

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

$$\begin{aligned}\tilde{L}_{k+1}(v_{k+1}, v_k) \approx & \frac{v_k^{(1)} - v_{k+1}^{(0)}}{h_k} \cdot (\frac{h_k}{2} + \frac{h_{k+1}}{2}) + \frac{v_k^{(2)} - v_{k+1}^{(0)}}{h_k} \cdot (\frac{h_k}{2} + \frac{h_{k+1}}{2}) \\ & + \frac{v_{k+1}^{(3)} - v_{k+1}^{(0)}}{h_{k+1}} \cdot (\frac{h_k}{2} + \frac{h_{k+1}}{2}) + \frac{v_{k+1}^{(3)} - v_{k+1}^{(0)}}{h_{k+1}} \cdot (\frac{h_k}{2} + \frac{h_{k+1}}{2}) \\ = & \frac{3}{4}(v_k^{(1)} + v_k^{(2)} - 2v_{k+1}^{(0)}) + \frac{3}{2}(v_{k+1}^{(3)} + v_{k+1}^{(5)} - 2v_{k+1}^{(0)}) \end{aligned} \tag{3.16}$$

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

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

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.

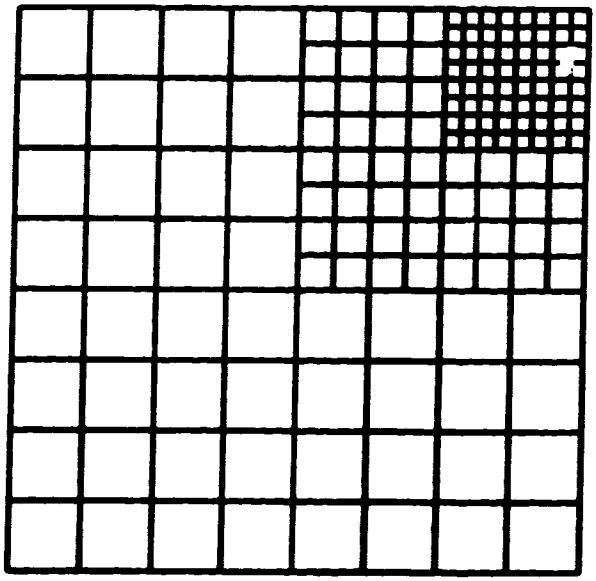


Figure 4.1

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 covergence 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 $\|v_k - u^*\|$
3×3	1	2	24	-
5×5	2	3	121	-
9×9	3	4	654	0.706677e-1
9×9	4	4	829	0.160698e-1
9×9	5	4	1017	0.434065e-2

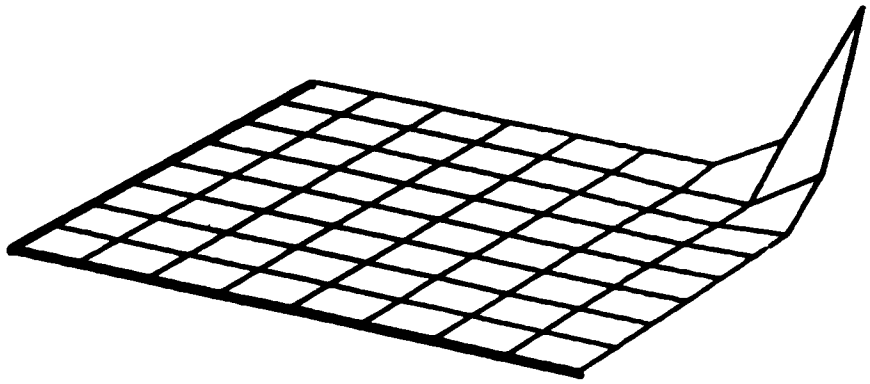
Table 4.1

Local mesh refinement algorithm on model problem ($\alpha = 10^2$)

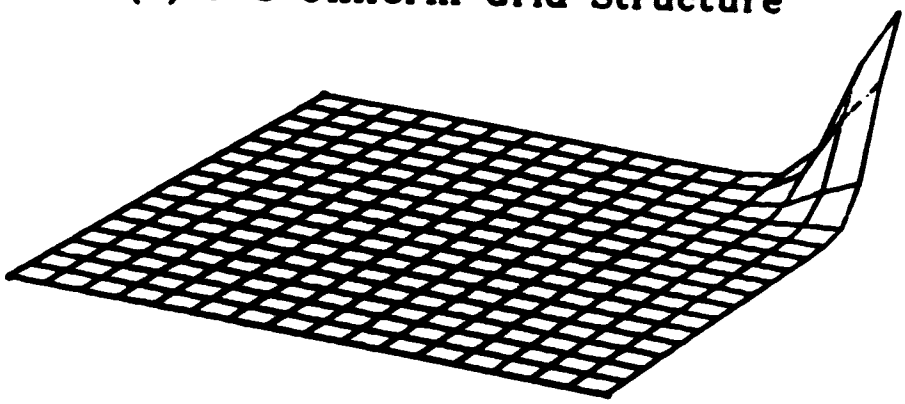
No. of Grid Points	No. of Grid Level	No. of Iterations	CPU (millisec.)	Error $\|v_k - u^*\|$
3×3	1	2	24	-
5×5	2	3	121	-
9×9	3	4	654	0.706677e-1
17×17	4	4	3485	0.160657e-1
33×33	5	4	15651	0.429657e-2

Table 4.2

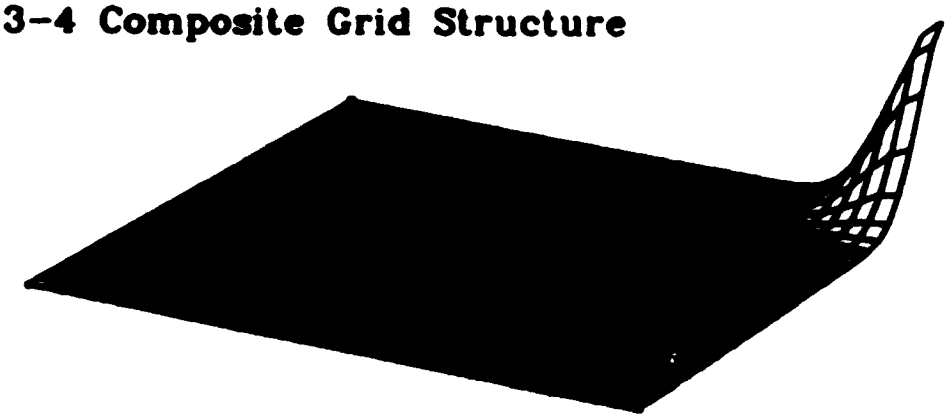
Uniform multi-grid method on model problem ($\alpha = 10^2$)



(a) 3-3 Uniform Grid Structure



(b) 3-4 Composite Grid Structure



(c) 3-5 Composite Grid Structure

Figure 4.2

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 960, 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 8-13, 1984, Notes on Numerical Fluid Mechanics, Vol.11, Viewag, Braunschweig, 1984.

[6] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31(1977), pp.333-390.

[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.277-318.

[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. Foerster and K. Witsch, *Multi-grid software for the solution of elliptic problems on rectangular domains: MGOO (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 960, 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., Boole 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. Jesperson, *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. Böhmer 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.

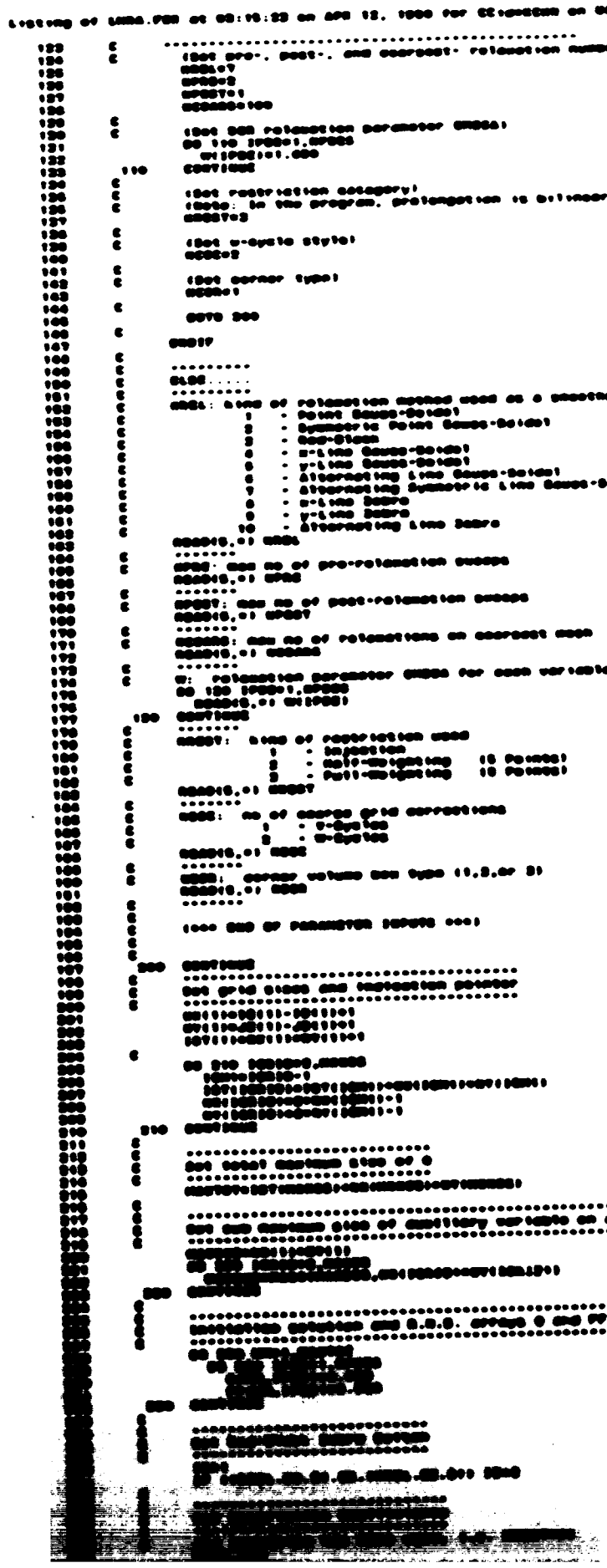
[20] S. Sivaloganathan and G. J. Shaw, *A multi-grid method for recirculating flows*, Internat. J. Numer. Methods Fluids, Vol.7, 1987, pp.417-440.

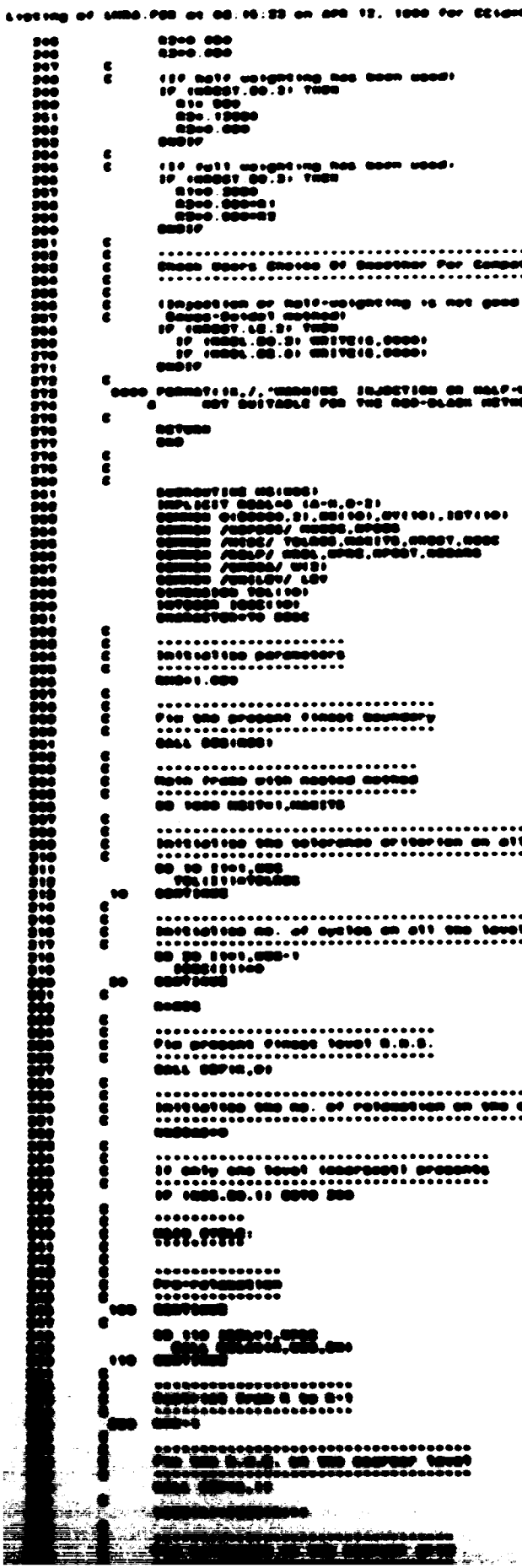
[21] D. M. Young and R. T. Gregory, *A survey of numerical mathematics, Vol.1&2*, Addison-Wesley, 1972.

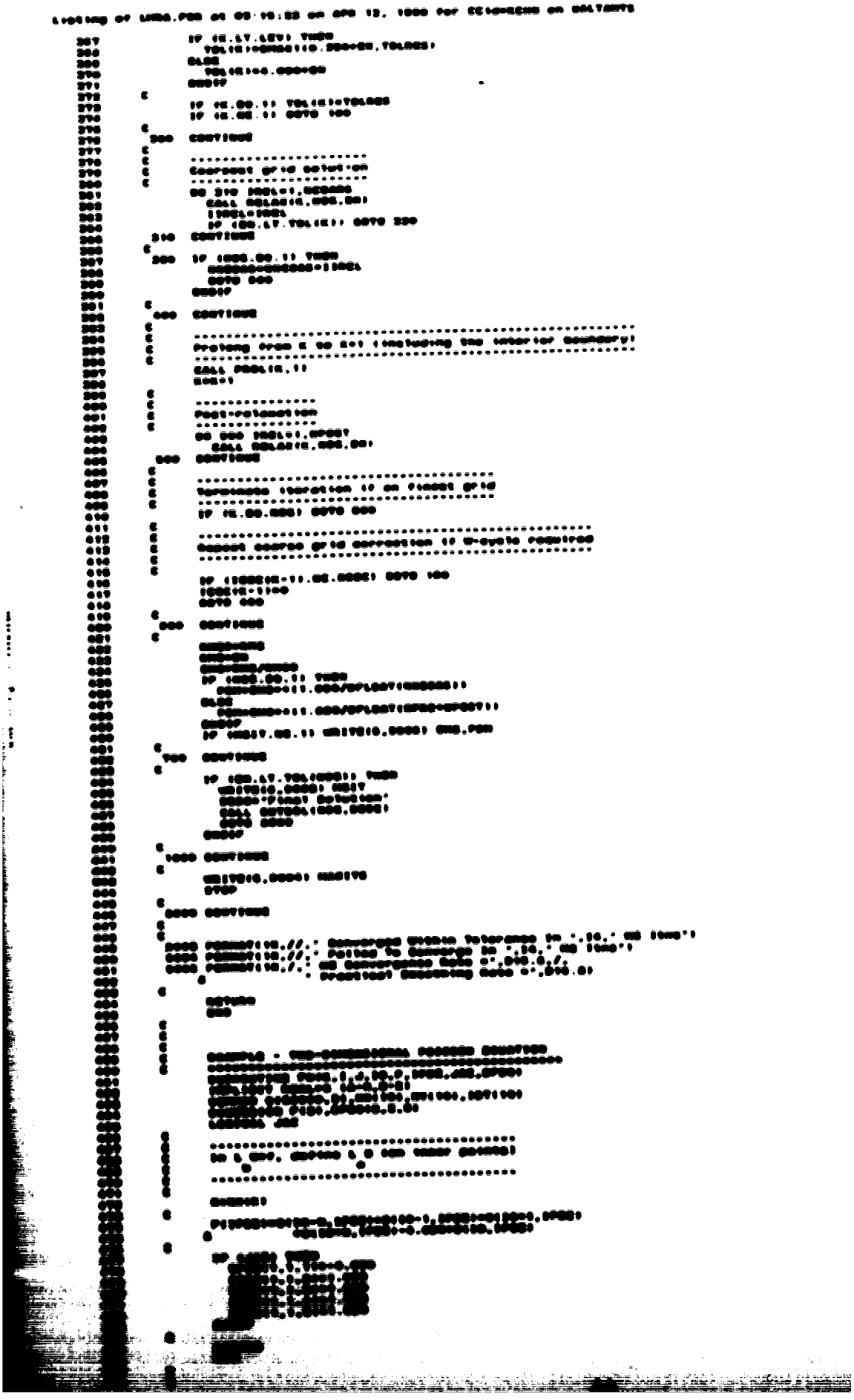
[22] T. A. Zang, Y. S. Wong and M. Y. Hussaini, *Spectral multi-grid methods for elliptic equations*, J. Comput. Phys., 48(1982), pp.485-501.

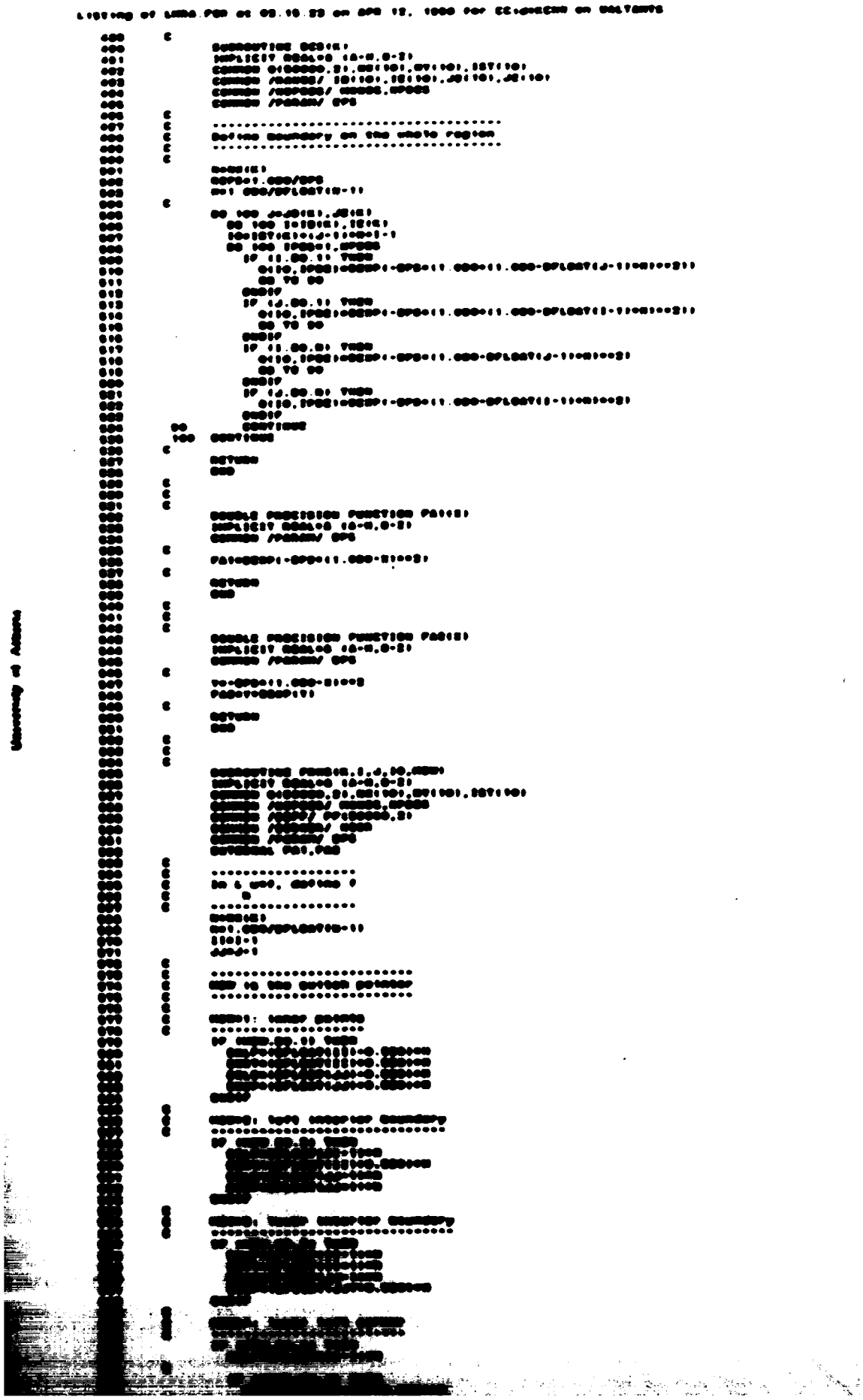
[23] T. A. Zang, Y. S. Wong and M. Y. Hussaini, *Spectral multi-grid methods for elliptic equations II*, NASA-CR-172131, ICASE, NASA Langley Research Center, 1983.

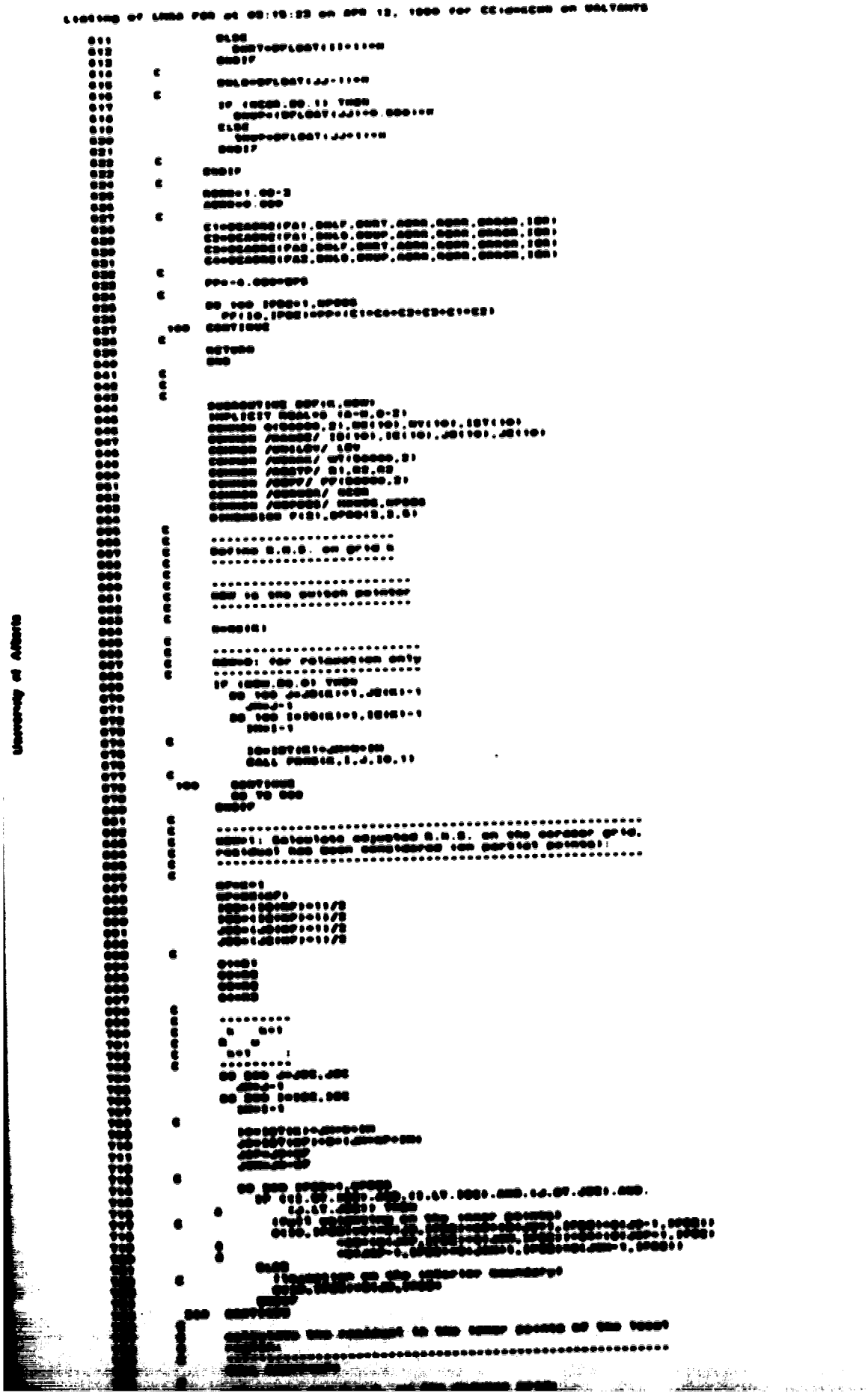
Appendix: List of a FORTRAN77 Program for LMRA

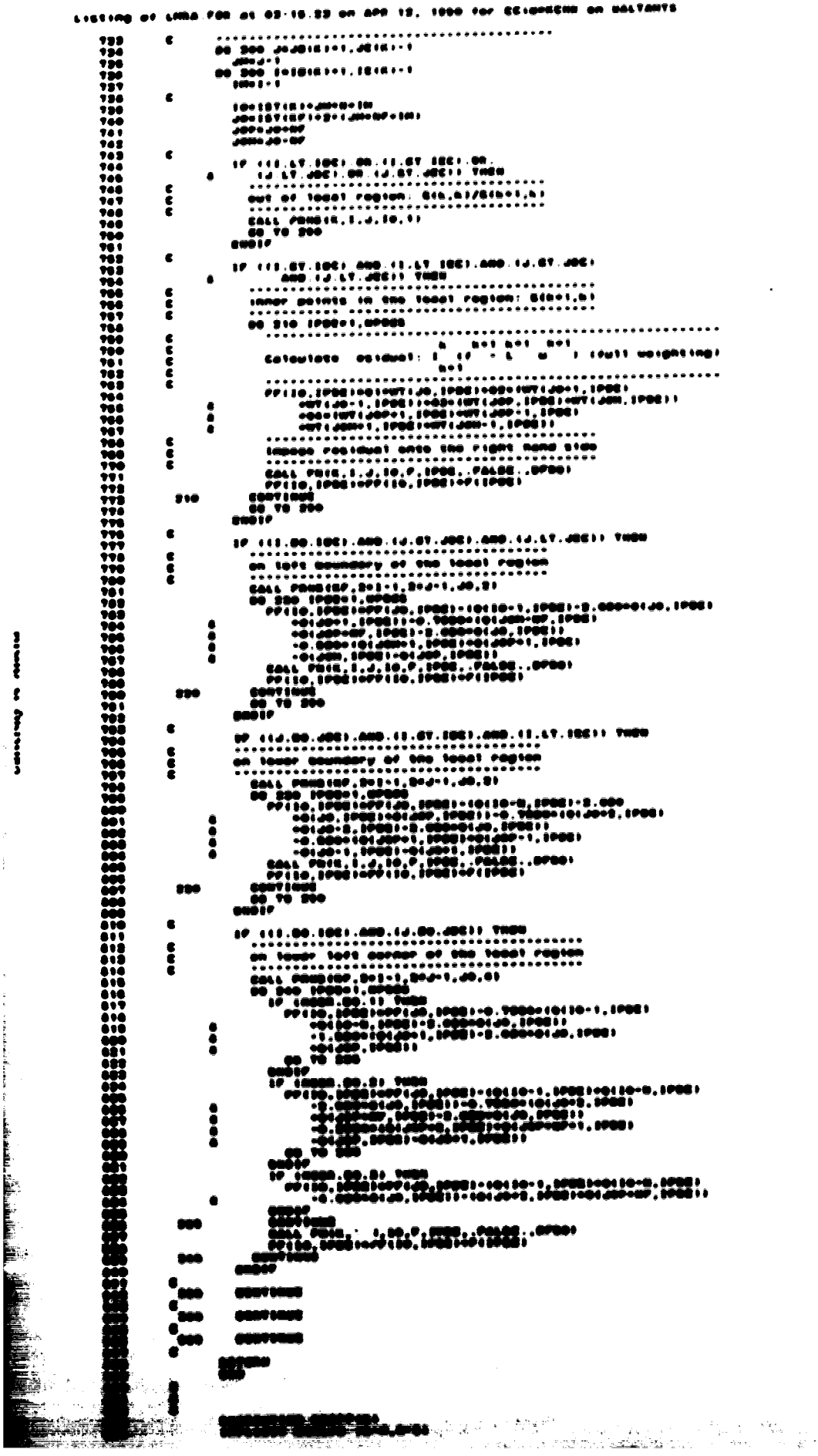


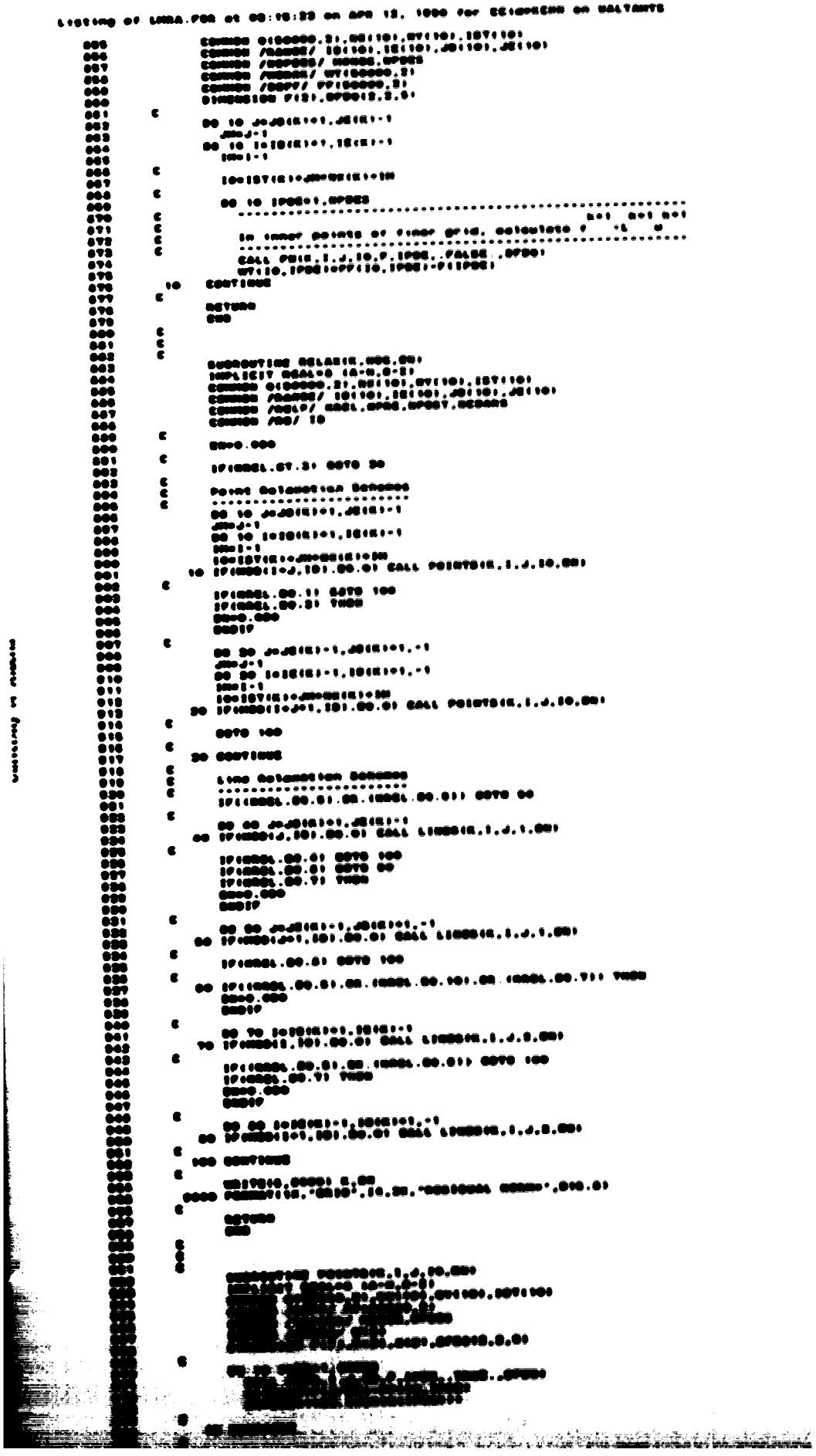


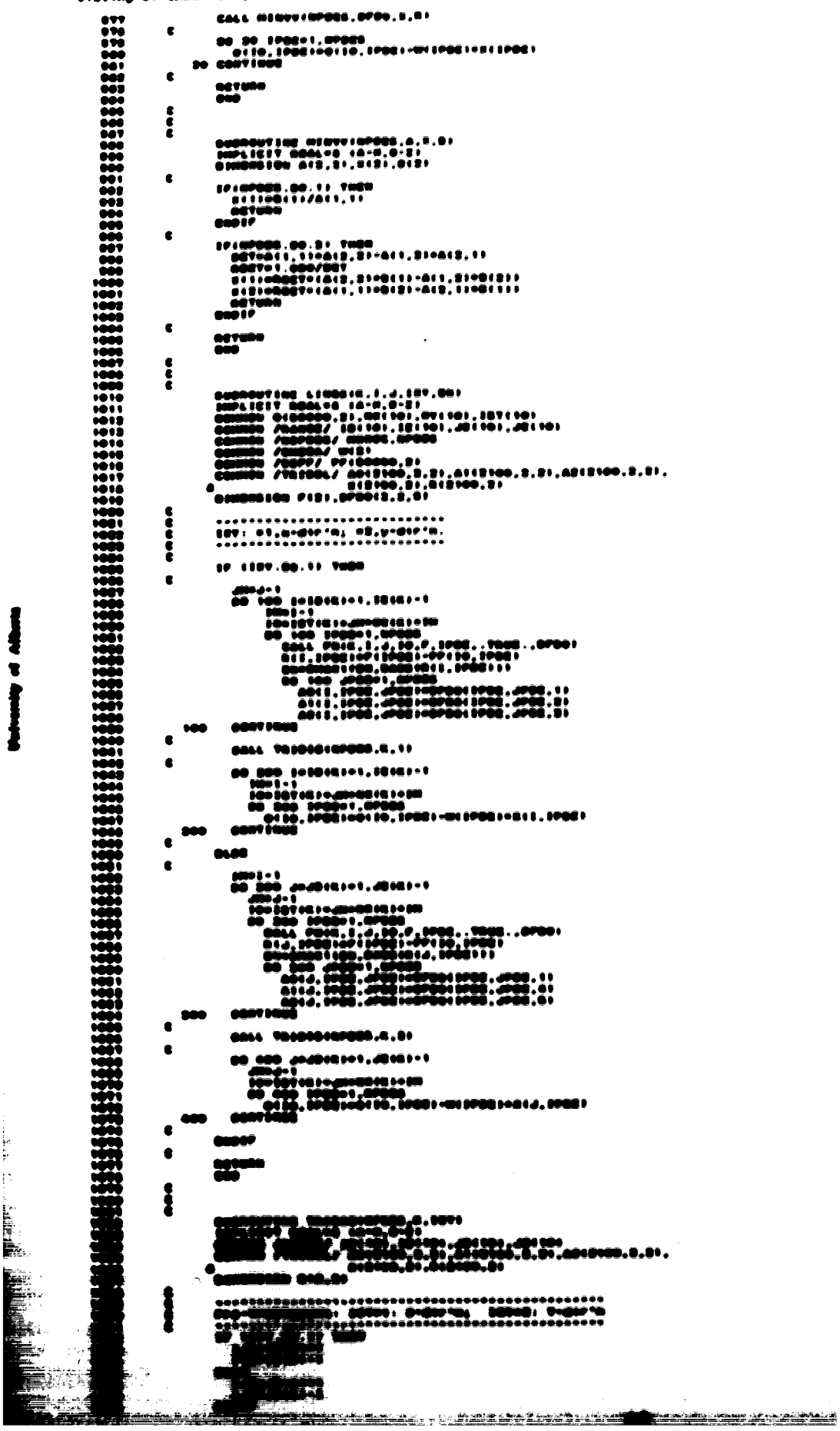


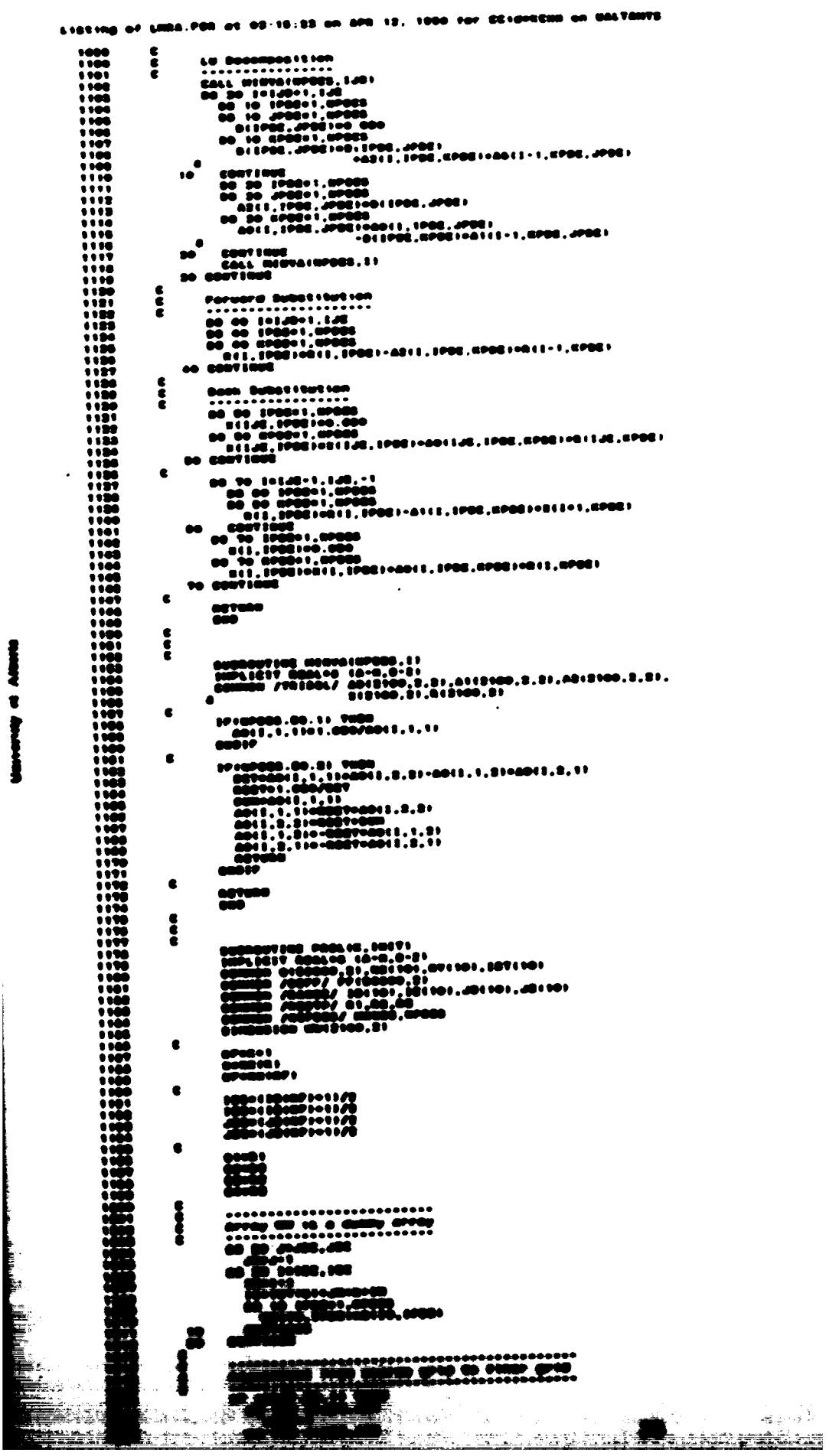












Listing of LUDA.FOR at 08:16:33 on APR 12, 1990 for CC-[illegible] on [illegible]

```
C
          WRITE(6,920) I,J,EXACT(K0,IPDE),U(I0,IPDE),ERR(K0,IPDE)
   10   CONTINUE
  100 CONTINUE
C
      WRITE(6,930) K,ERR
C
  900 FORMAT(1X,/,'RESULTS ON LEVEL ',I2,' IS:')
  910 FORMAT(1X,/,5X,'I',5X,'J',10X,'EXACT',12X,'U(I,J)',12X,'ERROR')
  920 FORMAT(1X,/,2(I5,1X),3X,3(D16.8,3X))
  930 FORMAT(1X,/,'THE MAXIMUM ERROR ON LEVEL ',I2,' IS  ',D16.8)
C
      RETURN
      END
```