

# CONTRACTOR REPORT

SAND96-2343  
Unlimited Release  
UC-706

## An Adaptive Gridless Methodology in One Dimension

N. Todd Snyder, Christine E. Hailey  
Mechanical and Aerospace Engineering  
Utah State University  
Logan, UT 84322-4130

Prepared by  
Sandia National Laboratories  
Albuquerque, New Mexico 87185 and Livermore, California 94550  
for the United States Department of Energy  
under Contract DE-AC04-94AL85000

Approved for public release; distribution is unlimited.

Printed September 1996

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

**NOTICE:** This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from  
Office of Scientific and Technical Information  
PO Box 62  
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from  
National Technical Information Service  
US Department of Commerce  
5285 Port Royal RD  
Springfield, VA 22161

NTIS price codes  
Printed copy: A05  
Microfiche copy: A01

SAND 96-2343  
Unlimited Release  
Printed September 1996

Distribution  
Category UC-706

**AN ADAPTIVE GRIDLESS METHODOLOGY  
IN ONE DIMENSION**

N. Todd Snyder  
Christine E. Hailey  
Mechanical and Aerospace Engineering  
Utah State University  
Logan, UT 84322-4130

Sandia Contract AK-2508

**ABSTRACT**

Gridless numerical analysis techniques offer great potential for accurately solving for flow about complex geometries or moving boundary problems. Because gridless methods do not require the point connectivity associated with traditional techniques, the mesh cannot twist or distort. The gridless method utilizes a Taylor series about each point to obtain the unknown derivative terms from the current field variable estimates. The governing equation is then numerically integrated to determine the field variables for the next iteration. The effects of point spacing and Taylor series order on accuracy are studied, and they follow similar trends of traditional numerical techniques. Introducing adaption by point movement using a spring analogy allows the solution method to track a moving boundary. The adaptive gridless method models linear, nonlinear, steady, and transient problems. A comparison with known analytical solutions is given for these examples. Although point movement adaption does not provide a significant increase in accuracy, it helps capture important features and provides an improved solution.

## PREFACE

This report is the final deliverable to Sandia National Laboratories (SNL) for Contract Number AK-2508 between SNL and Utah State University. It describes the results for Tasks Three and Four. Results for Tasks One and Two were presented in a previous report (Wolfe, et al., 1996).

The report is largely the master's thesis of N. Todd Snyder and represents a fairly general study of adaption in one dimension using gridless methodology. Embedded within the broader study are the results for Tasks Three and Four. Specifically, the results of Task Three are presented in Chapter Eight and the results of Task Four are presented in Chapter Four.

The authors thank James M. Nelsen for the initial code and for his insight into adaptive methods, and also thank Walter P. Wolfe for his guidance throughout this program.

# CONTENTS

1.	OVERVIEW OF GRIDLESS TECHNIQUES .....	1
	The Need for Gridless Methods .....	1
	Traditional Solution Techniques .....	1
	Solving the Governing Partial Differential Equations .....	2
	The Gridless Computational Method of Liszka and Orkisz .....	3
	Code Modularity .....	6
	The Purpose of Adaption .....	6
	Goals for Implementing the Adaption Algorithm into the Gridless Method .....	7
	Adaption Methodology .....	7
	Outline for Developing and Analyzing the Adaptive Gridless Methodology .....	9
2.	STEADY-STATE GRIDLESS METHODOLOGY WITHOUT ADAPTION .....	11
	Introduction .....	11
	Determining the Spatial Derivatives .....	11
	Updating the Dependent Variable .....	12
	The Integration Time Increment .....	12
	Residual Definition .....	13
	Error Definition .....	13
	Convergence Definition .....	14
	Test Case One .....	14
	Test Case Two .....	16
	Test Case Three .....	19
	Summary .....	21
3.	DEVELOPMENT OF SUGGESTED DEFAULT PARAMETERS .	23
	Introduction .....	23
	The Number of Neighboring Cloud Points .....	24
	The Taylor Series Approximation Order (p-adaption) .....	24
	The Number of Mesh Points (h-adaption) .....	27
	Summary .....	28

4.	PROBLEM CONDITIONING .....	29
	The Matrix Condition Number .....	29
	Number of Neighboring Cloud Points .....	31
	Total Number of Points (Spacing) .....	33
	Taylor Series Order .....	33
	Guaranteed Digits of Accuracy .....	34
	Summary .....	34
5.	ADAPTION BY POINT MOVEMENT .....	37
	When to Begin Adaption .....	37
	Adaption by Point Movement .....	38
	When Adaption is Necessary .....	40
	Selecting an Appropriate Point Movement Parameter .....	41
	Point Movement Algorithm Performance .....	42
	Point Movement Ability to Solve Equation (10) Regardless of the Initial Point Seeding .....	43
	Point Movement Ability to Solve Equation (14) .....	44
	Summary .....	45
6.	THE ADAPTION COEFFICIENT .....	47
	Introduction .....	47
	The Traditional Spring Analogy .....	47
	Summary .....	50
7.	ADAPTION BY MESH REFINEMENT .....	51
	Introduction .....	51
	When Adaption by Refinement is Necessary .....	51
	Sequence of Adaption Methods .....	51
	Adding Points .....	53
	Performance of the Combined Refinement and Movement Adaption Methods .....	53
	Summary .....	56

8.	TRANSIENT SOLUTIONS USING THE GRIDLESS METHOD . .	57
	Introduction . . . . .	57
	Extending the Solution Method for Solving Transient Problems . . . .	57
	A Simple Transient Test for the Gridless Method . . . . .	58
	Numerical Modeling of a Transient Problem with a Time-Periodic Boundary Condition . . . . .	59
	A Moving Boundary Solution . . . . .	61
	Grid Prediction . . . . .	63
	Summary . . . . .	64
9.	GLOBAL ADAPTION . . . . .	65
	Introduction to Global Adaption . . . . .	65
	The Global Adaption Method . . . . .	65
	Global Adaption Performance for Steady-State Problems . . . . .	67
	Summary . . . . .	69
10.	CONCLUSIONS . . . . .	71
	The Strengths of the Gridless Method . . . . .	71
	Purpose of Adaption . . . . .	71
	Goals for Implementing Adaption . . . . .	71
	Multidimensional Gridless Adaption . . . . .	72
	Summary . . . . .	73
	REFERENCES . . . . .	75

**Figures**

1	Graphical relation between the cloud points and the focus point . . . . .	4
2	Initial point distribution and function values . . . . .	12
3	Residual and average error behavior for equation (10) . . . . .	15
4	Solution to equation (10) . . . . .	16

5	Residual and average error behavior for equation (12) . . . . .	17
6	Solution to equation (12) . . . . .	18
7	Derivatives of the solution to equation (12) . . . . .	18
8	Residual and average error behavior for equation (14) . . . . .	20
9	Solution to equation (14) . . . . .	21
10	Derivatives of the solution to equation (14) . . . . .	26
11	Balancing computation time and error . . . . .	25
12	Effect of the number of points on final error for equation (12) . . . . .	27
13	Condition number vs. number of cloud points . . . . .	31
14	Condition number vs. number of points . . . . .	34
15	Condition number vs. Taylor series order . . . . .	34
16	Guaranteed digits of accuracy for a given Taylor series order . . . . .	35
17	Repositioning of an initially clustered point distribution . . . . .	43
18	Residual and error comparisons with adaption for equation (10) . . . . .	44
19	Solution to equation (14) with point movement adaption . . . . .	45
20	Final error for equation (10) with variable stiffness weightings . . . . .	49
21	Final error for equation (12) with variable stiffness weightings . . . . .	50
22	Solution to equation (10) with full adaption . . . . .	54
23	Solution to equation (12) with full adaption . . . . .	55
24	Solution to equation (14) with full adaption . . . . .	55
25	Solution to equation (36) at various times . . . . .	58

26	Solution to equation (38) at various times .....	60
27	Solution to equation (38) at various times with relaxed adaption .....	62
28	Error comparison for two solutions of equation (38) .....	62
29	A moving boundary solution for equation (36) .....	63
30	Error comparison for standard and prediction adaption methods .....	64
31	Solution to equation (12) with global adaption .....	68
32	Error comparison for local and global solutions to equation (12) .....	68
33	Solution to equation (36) with global adaption .....	70
34	Error comparison for local and global solutions to equation (36) .....	70

**Tables**

1	Error Matrix for the Taylor Series Order and Cloud Point Number ...	26
2	Error Matrix for the Total Number of Mesh Points .....	27
3	Condition Numbers .....	32

Blank Page

## CHAPTER 1

# OVERVIEW OF GRIDLESS TECHNIQUES

### The Need for Gridless Methods

Numerous important fluid and structure problems require tracking a moving membrane responding to the pressure field induced by the motion of a fluid. Examples include the deployment and inflation process of a parachute or an airbag; the aeroelastic phenomenon known as aileron buzz; and the opening and closing of pressure-driven heart valves and angioplasty balloon devices. Safety issues are also a concern in areas where a rapid decompression may occur, such as a tire blowout at highway speed, or a sudden tear followed by a rapid pressure loss in lighter-than-air vehicles.

These fluid/structure problems challenge existing numerical techniques. The solution algorithm, in addition to solving for the fluid dynamic field variables, must also provide for the motion of both the membrane surface and the surrounding mesh. Besides being computationally expensive, this transient grid remeshing may often cause the grid to distort and twist into undesirable shapes. Convergence and solution accuracy degrade appreciably when the motion of the adapted grid results in a highly distorted mesh. Typically, such grid-related errors occur where large-scale structural deformation or large flow gradients occur. It is precisely in these regions where the greatest accuracy is desired.

Gridless technology may alleviate the problems associated with grid generation and grid distortion that occur during the adaption process. The strength of this methodology lies in its ability to model any geometry in one, two, or three dimensions using “clouds” of discrete points. These points are not required to be connected to form a grid as in conventional computational fluid dynamics (CFD) and structural mechanics algorithms. Since a point is not permanently connected to any other point, the distribution of points cannot tangle or twist during the adaption process. This preserves the solution integrity, even when adapting to complex or moving geometries.

### Traditional Solution Techniques

Although considerable progress has been made in traditional grid-based solution methods, these methods all have the same common limitation. They have structure. A

mesh is structured if its connectivity is of finite difference type. In other words, if the grid may be represented by a mesh of cubic elements through a transformation of coordinates, the method is structured (George 1991). Structured grids typically consist of four-sided cells in two dimensions and six-sided cells in three dimensions. Because of the inherent geometric limitation of this cell shape, sophisticated meshing methodologies are required for modeling complex geometries. This, in turn, complicates the algorithm required to obtain a solution with the structured grid by complex coordinate transformations. This process requires a significant level of human time for generating the mesh, as well as the added computational time for the transformations. The complexity is further enhanced when the grid is required to model unsteady flow or moving boundaries.

A second approach uses unstructured grids. Unstructured grids do not depend on finite difference formulations (Batina 1992). Therefore, more efficient geometries may be considered, and coordinate transformations may not be required. In two dimensions, unstructured grids generally consist of triangular-shaped cells. Grids constructed from these shapes may be easily oriented to conform to very complex geometries. This allows solutions to be obtained over complex shapes without making changes to basic solution algorithms. However, application of this method in three dimensions requires the use of the inefficient tetrahedral shape. This results in an excessively large number of cells and a substantial increase in computational time. The problem may be alleviated by using the more efficient cell shape of a polyhedron with a triangular cross section. This, however, places structure back into the grid and again causes difficulty when modeling complex geometries (Batina 1992).

According to Batina, what is truly required to advance computational techniques is not to take a step backward toward grid structure, but to step forward and develop methods that do not require the use of grids at all (Batina 1993). In gridless techniques, only points, or more specifically clouds of points, are required. Therefore, gridless techniques offer the greatest potential for accurately and efficiently solving for flow about complex geometries, moving boundary problems, and problems involving the dynamic coupling of a fluid with a structure.

### **Solving the Governing Partial Differential Equations**

Since a gridless solution cannot depend on the connectivity between points, the governing equations must be solved discretely at each point rather than globally stepping through the grid. To determine the governing equations, the partial differential equations (PDEs) must be solved at every point in the domain individually. The derivatives

are estimated by relating information about each point with the corresponding information from the cloud of neighboring points surrounding the focus point.

Batina solved the governing partial differential equations directly by performing local least-squares curve fits in each cloud of points. He then analytically differentiated the resulting curve-fit equation to approximate the derivatives of the PDEs (Batina 1993). This method is neither a finite difference nor a finite volume approach since differences, metrics, lengths, or volumes are not computed. This method works well when only first-order derivative terms are required.

A second method recommended by Liszka and Orkisz uses a Taylor series expansion within each cloud of points to directly determine the PDE derivatives (Liszka and Orkisz 1980). This method follows more traditional CFD concepts, but does not require finite differences to be computed. Because of the ability to directly obtain higher order terms for the Taylor series and following more traditional techniques, this second method has been selected for this study and will be further outlined for the one-dimensional problem.

### The Gridless Computational Method of Liszka and Orkisz

A “cloud” of neighboring points is selected for each point in the problem domain. In one dimension, the cloud points are the nearest neighbors. The point around which the cloud is developed will be called the focus point. Note, in this report the cloud does not include the focus point. The relation of these points is given graphically in Figure 1.

Once the cloud of points that will influence each focus point has been determined, a Taylor series approximation is expanded from each focus point to the points in its surrounding cloud. The distance from the focus point to each of the neighboring cloud points and an estimate for the cloud point function values must be known.

$$f_i = f_0 + \sum_{n=1}^{\infty} \frac{(x_i - x_0)^n}{n!} \frac{d^n f_0}{d x^n}$$

where  $x_0 =$  the focus point position

$x_i =$  the position of cloud point  $i$

$$f_i = f(x_i)$$

$$f_0 = f(x_0)$$

(1)

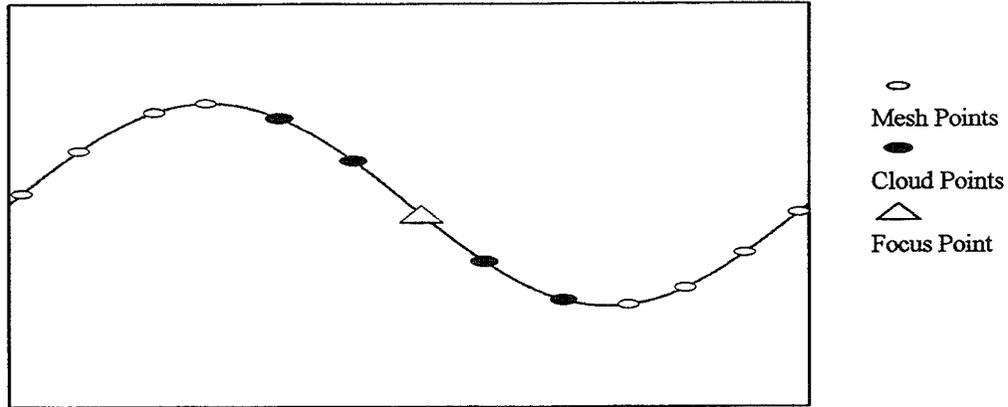


Figure 1. Graphical relation between the cloud points and the focus point.

Including only the first  $n$  terms of the Taylor's series, this expression may be written as:

$$h_i \left( \frac{df_0}{dx} \right) + \frac{h_i^2}{2!} \left( \frac{d^2f_0}{dx^2} \right) + \frac{h_i^3}{3!} \left( \frac{d^3f_0}{dx^3} \right) + \dots + \frac{h_i^n}{n!} \left( \frac{d^n f_0}{dx^n} \right) = (f_i - f_0) \quad (2)$$

where  $h_i = (x_i - x_0)$

The position and current approximation to the function value are known for both the focus point and cloud point. Therefore, the derivative values are the only unknowns in this equation. By writing this expansion for  $n$  cloud points, and including only the first  $n$  derivative terms of the Taylor series expansion, a linear set of  $n$  equation and  $n$  unknown is achieved. This may be written in matrix form as:

$$[A][Df] = [f]$$

$$[A] = \begin{bmatrix} h_1 & \frac{h_1^2}{2!} & \frac{h_1^3}{3!} & \dots & \frac{h_1^n}{n!} \\ h_2 & \frac{h_2^2}{2!} & \frac{h_2^3}{3!} & \dots & \frac{h_2^n}{n!} \\ h_3 & \frac{h_3^2}{2!} & \frac{h_3^3}{3!} & \dots & \frac{h_3^n}{n!} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_n & \frac{h_n^2}{2!} & \frac{h_n^3}{3!} & \dots & \frac{h_n^n}{n!} \end{bmatrix} \quad [Df] = \begin{bmatrix} \frac{df_0}{dx} \\ \frac{d^2f_0}{dx^2} \\ \frac{d^3f_0}{dx^3} \\ \vdots \\ \frac{d^nf_0}{dx^n} \end{bmatrix} \quad [f] = \begin{bmatrix} (f_1 - f_0) \\ (f_2 - f_0) \\ (f_3 - f_0) \\ \vdots \\ (f_n - f_0) \end{bmatrix} \quad (3)$$

This system of linear equations can easily be solved for the derivative elements at each location. The main difficulty in a successful application of such an approach is to avoid a singular or an ill-conditioned  $[A]$  matrix to obtain acceptable derivatives.

As noticed in the matrix equation, if the governing equations require first- and second-order derivatives, then only two cloud points are required—two Taylor series expansion equations to obtain the two desired unknowns. In order to improve the accuracy in approximating the derivatives, the number of cloud points may be increased. This over-determined system of linear equations can be solved by a least-squares approximation or using singular-value decomposition.

Once the spatial derivative values have been determined, this information is placed into the governing PDE. Using these new values, the governing equation is stepped forward in time using a Runge-Kutta numerical integration scheme. A steady-state solution is achieved when the change in the field variable values is small during the integration step. Transient problems require a sufficiently small time increment to maintain solution stability.

## Code Modularity

The gridless solution technique implies that each step in the process is distinct and self-contained. This separation in solution steps allows the numerical code to be written in a modular form. One module may contain the logic for selecting the points to include in the cloud of influence for each focus point, while another might determine how the mesh will adapt to provide a better solution. More important, the module that solves for the derivatives and the behavior of the field variables is independent of the governing equation's module. Therefore, changing the governing equations will not require any change in the actual code other than modifying the boundary condition and governing equations themselves.

## The Purpose of Adaption

Although a basic understanding of gridless methods can be gained from the literature, very little, if any, work has been done to apply adaption to these gridless techniques. The extension of these standard adaption techniques to gridless methods appears to be straightforward. In fact, adaption for gridless techniques should be easier since points are geometrically simpler structures than cells.

The purpose of adaption is to complement and enhance numerical methods. Effective adaption will provide a more accurate solution and minimize the computational effort. A more accurate numerical solution will globally approach the exact or true solution while capturing its important local features. Computational effort is minimized when calculations that provide negligible contributions to the accuracy or resolution of the solution are eliminated.

Adaption techniques can be categorized into three distinct types: movement, refinement, and higher order approximations. All three of these types have been utilized in traditional grid-based solvers. Adaption by point movement is the most mature of the three, with the spring analogy perhaps the most widely used implementation. In this technique, the mesh structure is moved according to a local spring constant, which is a pre-defined function of the solution field. Adaption by mesh refinement, sometimes referred to as h-adaption, will locally refine or coarsen the mesh by the addition or deletion of points (Gui and Babuška 1986). Adaption by higher order approximations, sometimes referred to as p-adaption, will locally modify the order of the approximating function to achieve the best solution (Demkowitz et al. 1989). Although adaption by higher order approximations or

mesh refinement offers the greatest possibility for improving accuracy, effective implementation of point movement is most critical for the gridless method. Point movement allows the method to track a moving boundary.

The primary consideration for adaption, regardless of the type, is the composition of the algorithm that drives the adaption procedure. This logic path must consider several relevant issues such as: when to adapt; the proper forcing function for adaption; the order and concurrency of adaption types; and perhaps most important, any imposed limitations on the adaption procedure. This is the primary focus of the research work to be outlined in this report.

### **Goals for Implementing the Adaption Algorithm into the Gridless Method**

The expected strengths of the gridless method lie in its ability to model any geometry without relying on the point connectivity associated with traditional techniques. The one-dimensional research must verify this concept and provide a background for future efforts in multidimensions. Ultimately, the gridless method should be capable of accurately modeling a three-dimensional transient boundary problem. Adaption is the key to realizing this objective. To simplify the method and allow for a smooth transition to multidimensional models, the following goals were outlined for implementing adaption into the gridless technique.

1. Achieve a level of semi-autonomy allowing the user to intervene only to prescribe general purpose accuracy parameters;
2. Incorporate generality to permit the solution of a wide class of problem sets;
3. Achieve a state of robustness in which the user-defined level of accuracy can be accomplished regardless of the initial point seeding;
4. Allow generality of the adaption algorithm technology to accommodate multi-dimensional problems.

### **Adaption Methodology**

Determining the appropriate parameters on which to adapt is the primary consideration in constructing the adaption algorithm. Because of their generality and their availability with this gridless technique, the adaption algorithm utilizes the dependent variable, the gradient, and the second derivative. This departs slightly from adaption in many commercial

solvers, which only use the gradient of the dependent variable (Anderson, Tannehill, and Pletcher 1984). By allowing differences in both the variable magnitude and its first and second derivatives (changes in the slope and curvature of the solution) to influence the adaption of points, the methodology is both generalized and moderated.

To determine when adaption is necessary, two adaption (spring) coefficients ( $k_{i-1}, k_{i+1}$ ) are calculated for each point. These coefficients compare the dependent function value and derivative values with similar values at neighboring points. Each of the three components is uniformly weighted. To keep the magnitude of each component similar, the change in the values is divided by the total range.

$$\begin{aligned}
 k_{i-1} &= \left( \left| \frac{f(x)_i - f(x)_{i-1}}{(f(x)_{\max} - f(x)_{\min})} \right| + \left| \frac{f'(x)_i - f'(x)_{i-1}}{(f'(x)_{\max} - f'(x)_{\min})} \right| + \left| \frac{f''(x)_i - f''(x)_{i-1}}{(f''(x)_{\max} - f''(x)_{\min})} \right| \right) \\
 k_{i+1} &= \left( \left| \frac{f(x)_i - f(x)_{i+1}}{(f(x)_{\max} - f(x)_{\min})} \right| + \left| \frac{f'(x)_i - f'(x)_{i+1}}{(f'(x)_{\max} - f'(x)_{\min})} \right| + \left| \frac{f''(x)_i - f''(x)_{i+1}}{(f''(x)_{\max} - f''(x)_{\min})} \right| \right)
 \end{aligned} \tag{4}$$

For local adaption, the first adaption procedure implemented is point movement. The basic principle for point movement is a spring analogy in which the adaption spring coefficient ( $k$ ) times the distance to the next point remains constant. When the change in this relation is not within a specified tolerance, a point must be moved. The point movement process continues until all points satisfy the tolerance and the spring system achieves equilibrium.

$$|k_{i-1}(x_i - x_{i-1}) - k_{i+1}(x_{i+1} - x_i)| < tolerance \tag{5}$$

Similarly, we can assess the need for the addition or deletion of points. If the magnitude of any of the individual differences on both sides of a center point exceeds a user-defined upper bound refinement parameter, then a point should be added. Conversely, if the magnitude of all the individual differences declines below a user-defined lower bound refinement parameter, a point should be deleted from the system.

## **Outline for Developing and Analyzing the Adaptive Gridless Methodology**

The two underlying objectives for this research into an adaptive one-dimensional gridless method are first, to verify the capabilities of the gridless method to adapt and provide accurate solutions for a variety of problem types; and second, to provide background information for implementing a multidimensional adaptive gridless method. The intent of this work centers on investigating a variety of problem types, mesh distributions, and adaption variations. Ultimately, the method should model a moving boundary problem. The capabilities of each of these combinations are verified and compared in a working numerical model. Although many suggestions and trade-offs are discussed, no attempt has been made to find an optimal one-dimensional gridless solution technique.

Chapter 2 describes the numerical gridless model in greater detail. It develops the basic gridless method without adaption for solving one-dimensional problems. Comparisons of numerical solutions with known analytical solutions verify the capabilities of this model.

In Chapter 3, adaption by mesh refinement and higher order approximations are investigated individually. This chapter also considers time constraints and the optimum number of cloud points. Suggested default values for the Taylor series order and number of cloud points are developed.

Problem conditioning and stability are discussed in Chapter 4. As the Taylor series order increases, the numerical problem for the gridless method quickly becomes ill-conditioned. The effects of problem conditioning on solution accuracy are presented. The effect of point spacing (location) on problem conditioning is also presented.

Chapter 5 introduces adaption by point movement into the gridless technique using a spring analogy. To achieve goal one, the adaption algorithm is automated except for the input of a user-defined point movement parameter. The method's behavior and ability to adapt for a diverse range of function types, including linear and non-linear examples, are observed. The ability to achieve the same solution regardless of initial point seeding is verified, fulfilling goal three.

Chapter 6 lightly investigates the adaption stiffness coefficient and the spring analogy method. Although this coefficient is formed arbitrarily, some factors may provide more valuable information than others. To a large extent, the best adaptive stiffness coefficient is

problem dependent. However, several variations and the effects of these variations on accuracy are presented.

Adaption by mesh refinement is introduced and developed in Chapter 7. Solutions from the gridless method with adaption by refinement and point movement are compared with known analytical solutions for linear and nonlinear problems. The ability to achieve approximately the same mesh distribution regardless of the initial number of points is verified to further complete goal three.

In Chapter 8, the gridless code is extended to model transient problems. The ability of the model to maintain time accuracy for various transient problems is observed. A predictor method is introduced, and its effect on time accuracy is discussed.

Chapter 9 introduces a novel global adaption methodology. This method significantly decreases the number of adaption calculations required. Solutions from this technique are compared with solutions of the local adaption methods and analytical solutions.

Finally, Chapter 10 summarizes the capabilities of the gridless technique with full adaption for any one-dimensional problem. Advantages and disadvantages of the method are discussed, and the most logical steps for implementing similar logic for adaption in two dimensions are presented.

## CHAPTER 2

# STEADY-STATE GRIDLESS METHODOLOGY WITHOUT ADAPTION

### Introduction

The initial point distribution and initial dependent variable estimates used by the program depend only on the boundary conditions. They are independent of the governing equation. The mesh points are distributed evenly between the left and right boundaries, while the initial function value at each of these points is determined from a linear interpolation between the boundary conditions. Using twenty-five total points, Figure 2 demonstrates the initial point distribution and initial dependent variable estimates given the boundary conditions  $f(0) = 1$  and  $f(1) = 0$ .

### Determining the Spatial Derivatives

As outlined in Chapter 1, this gridless technique revolves around solving the set of linear equations  $[A][Df] = [f]$  at each focus point for the unknown derivatives. Each equation in this set results from expanding a Taylor series at a focus point to one of the cloud points. These cloud points are generally the nearest neighbors. However, at least one cloud point must be on each side of the focus point. Each time the grid adapts, a new set of cloud points must be found.

To solve for the unknown spatial derivatives, the  $[A]$  matrix and  $[f]$  vector must be formed. Since the  $[A]$  matrix depends only on the spatial coordinate of each point, each iteration may use the same  $[A]$ . A new  $[A]$  matrix must be formed only after the grid adapts.

The first  $[f]$  vector forms from initial dependent variable estimates such as those in Figure 2.  $[f]$  represents the difference in the function values between the focus point and each of the cloud points. Since these values change and hopefully become more accurate with each iteration,  $[f]$  must be recalculated with each iteration.

With  $[A]$  and  $[f]$  known, the system  $[A][Df] = [f]$  may be easily solved for the partial derivative elements  $[Df]$ . For this task, the Fortran code utilizes a Gaussian elimination technique.

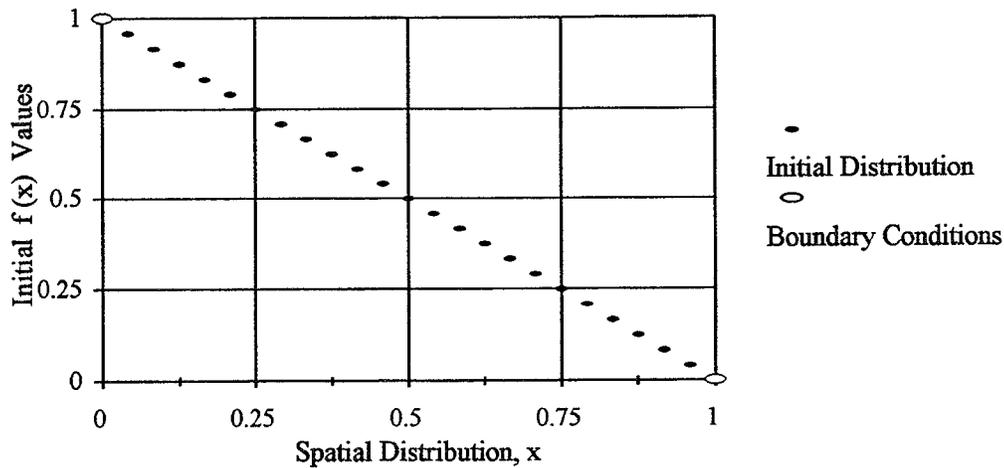


Figure 2. Initial point distribution and function values.

### Updating the Dependent Variable

To update the function values between iterations, the steady-state governing differential equation is set equal to zero and integrated numerically using a second-order Runge-Kutta integration scheme. A pseudo-time step ( $\Delta t$ ) is used as the Runge-Kutta increment. The Taylor series expansion previously discussed provides the current spatial derivative estimates. Numerically integrating the governing equation with the current estimated spatial derivative values achieves a more accurate estimate of the dependent variable value at each point.

The next iteration proceeds with these updated function values. The set of equations  $[A][Df] = [f]$  is solved for new spatial derivative estimates utilizing the latest function values. Placing these spatial derivatives in the governing equation and numerically integrating gives the next estimate of the dependent variable values. The process continues until the dependent variable values no longer change during this integration step and the steady-state governing equation is satisfied.

### The Integration Time Increment

The primary limiting factor on the number of iterations required before the solution converges and on the stability of the integration rests with the integration time step. The

time increment ( $\Delta t$ ) used during the Runge-Kutta integration depends on the minimum spacing between two points over the spatial domain ( $\Delta x_{\min}$ ) and the order of the approximating Taylor series. Until the mesh adapts, this increment will remain constant.

$$\Delta t = \frac{(\Delta x_{\min})^2}{2 \text{ Order}} \quad (6)$$

Although this time step seems conservative, it provided sufficient stability to solve all of the test cases. For most problems, a larger time step may be adequate.

### Residual Definition

To assess the performance of the gridless algorithm without adaption, several computational experiments were conducted. As previously described, the initial dependent variable distribution was found from linear interpolation between the boundary conditions. A normalized average residual ( $R$ ) measured the rate of convergence. This average residual is defined as the root-sum-square of the local residuals divided by the current total number of points times the residual of the initial iteration ( $R_0$ ).

$$R = \frac{\sqrt{\sum_{i=1}^{\text{number of points}} (f(x)_{i,k} - f(x)_{i,k-1})^2}}{R_0 \cdot (\text{number of points})} \quad (7)$$

*i* represent the point location  
*k* represent the current iteration number  
 $R_0$  is the residual of the initial iteration

### Error Definition

Since the exact analytical solution is known for each of the test problems, the local error is defined as the difference between the numerical and analytical values of the dependent variable. The average error ( $E$ ) is defined as the root-sum-square of the local errors divided by the current total number of points. This value is not normalized.

$$E = \frac{\sqrt{\sum_{i=1}^{\text{number of points}} (f(x)_{\text{exact}} - f(x)_{\text{numerical}})^2}}{\text{number of points}} \quad (8)$$

### Convergence Definition

The gridless code considers the governing equation solved when the residual no longer changes significantly over several iterations. In other words, a solution is converged when the sum of the residual (R) over ten iterations is less than a user-defined convergence parameter ( $\epsilon$ ) times the current time step. Typical values for the convergence parameter range from 0.0001 to 0.001.

*The governing equation is solved when:*

$$\Delta t_k \epsilon > \sum_{i=k-10}^k R_i \quad (9)$$

*i is the iteration number  
k is the current iteration*

The residual magnitude depends on the current time increment. As the grid adapts, the time increment will change. Different time increments could vary the magnitude of the residual. Therefore, a small residual alone cannot determine convergence. Including the time increment in the convergence criteria eliminates its dependence on point spacing.

### Test Case One

The first computational experiment tested the following simple linear, second order ordinary differential equation (ODE):

$$\frac{d^2 f(x)}{dx^2} - 5 \sin(\pi x) = 0 \quad (10)$$

Using the Dirichlet boundary conditions  $f(0) = 1$  and  $f(1) = 0$ , this ODE has the closed form solution:

$$f(x) = -\frac{5}{\pi^2} \sin(\pi x) - x + 1 \quad (11)$$

The gridless technique solved this differential equation using a fourth-order Taylor series approximation with four cloud points. The mesh consisted of twenty-five uniformly spaced points across the domain  $[0,1]$ .

Figure 3 shows the behavior of the residual (R) and the average error (E) during the calculations. While the global residual continued to decrease steadily, the average error leveled at approximately 6500 iterations. The minimum average error was  $1.89 \times 10^{-7}$ .

Figure 4 exhibits a comparison of the exact solutions for the dependent variable, the first derivative, and the second derivative with the solutions generated by the gridless technique. To allow better visualization, the values are normalized to fall between  $[-1,1]$ . The solid lines represent the analytical solutions while the symbols represent the numerical solutions. As observed, the comparisons are excellent for this simple linear ODE.

Because the dependent variable experiences only slow, gentle changes, this differential equation does not provide a significant challenge to the gridless method. However, this

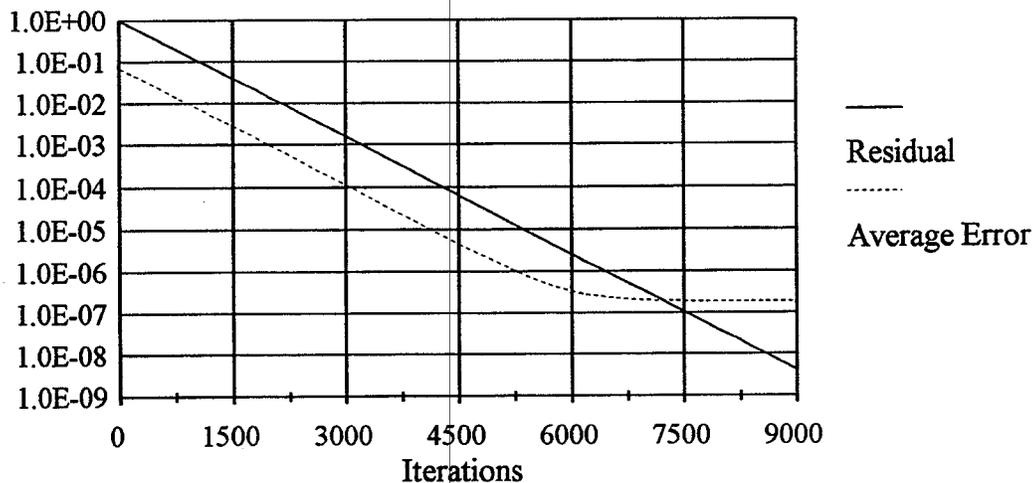


Figure 3. Residual and average error behavior for equation (10).

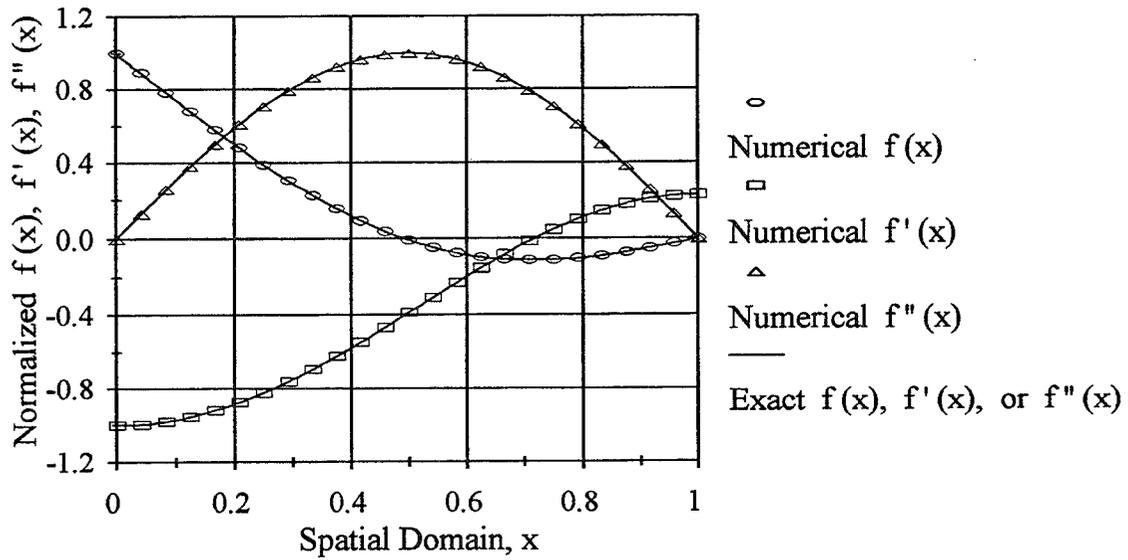


Figure 4. Solution to equation (10).

solution demonstrates the gridless methodology works and is capable of providing very accurate solutions.

### Test Case Two

The next equation tested was:

$$\frac{d^2f(x)}{dx^2} - (5\pi)^2 \sin(5\pi x) = 0 \quad (12)$$

Although equation (12) is similar to equation (10), it adds another degree of difficulty by compressing several oscillations into the domain  $[0,1]$ . Again, using the boundary conditions  $f(0) = 1$  and  $f(1) = 0$ , equation (12) has the closed form steady-state solution:

$$f(x) = -\sin(5\pi x) - x + 1 \quad (13)$$

The residual and the average error for this equation are given in Figure 5. The numerical solution utilized a fourth-order Taylor series with four cloud points. As in the

previous example, the global residual continues to decrease. However, the average error reaches a minimum after 127 iterations. It then increases to a steady-state average error of 0.001. This average error is considerably larger than the final error of the previous example. However, this is expected since the same number of total points is used to model a more complex function.

The residual and the average error for this equation are given in Figure 5. The numerical solution utilized a fourth-order Taylor series with four cloud points. As in the previous example, the global residual continues to decrease. However, the average error reaches a minimum after 127 iterations. It then increases to a steady-state average error of 0.001. This average error is considerably larger than the final error of the previous example. However, this is expected since the same number of total points is used to model a more complex function.

Figure 6 compares the numerical solution for equation (12) to the analytical solution from equation (13). The exact derivatives are compared to the numerical values of the first and second derivative in Figure 7. The solid lines represent the function's analytical solution or its derivatives while the symbols represent the numerical solutions.

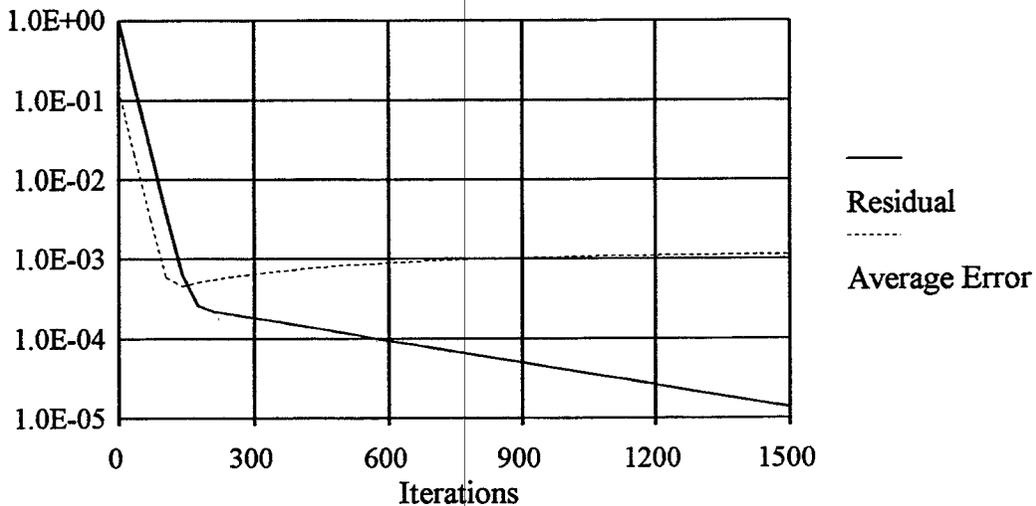


Figure 5. Residual and average error behavior for equation (12).

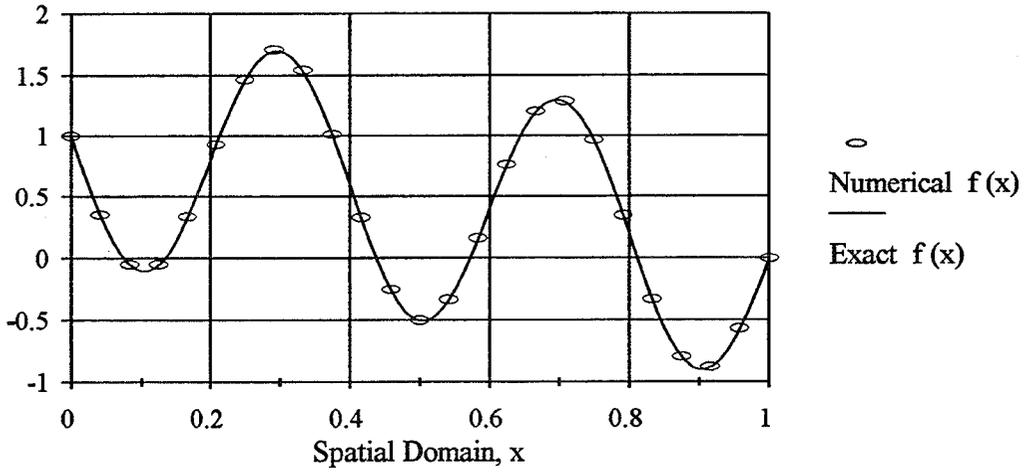


Figure 6. Solution to equation (12).

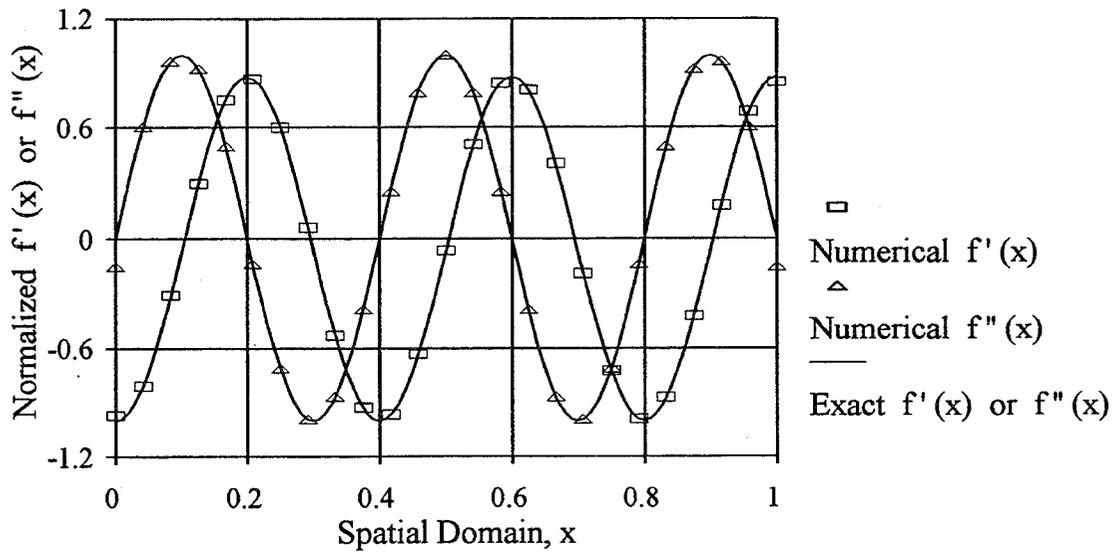


Figure 7. Derivatives of the solution to equation (12).

### Test Case Three

The final example considered is the nonlinear Burger's equation, given by equation (14).

$$\frac{\partial f(x,t)}{\partial t} + f(x,t) \frac{\partial f(x,t)}{\partial x} - \frac{1}{Re} \frac{\partial^2 f(x,t)}{\partial x^2} = 0 \quad (14)$$

*Re represents the Reynold's Number*

The nonlinear term in this equation will cause a discontinuity, or shock, to form in the solution, while the viscous term will tend to dissipate this effect. As the Reynolds number (Re) increases, the viscous term will decrease, causing an increase in the strength of the shock. To balance these two terms for this test, the Reynolds number was arbitrarily set at 50.

The closed form solution for this equation was obtained using a traveling wave solution technique that results in equation (15) (Logan 1987).

$$f(x,t) = \frac{f(-\infty,t) + f(\infty,t) e^{\left(\frac{Re}{2} (f(-\infty,t) - f(\infty,t))(x - ct)\right)}}{1 + e^{\left(\frac{Re}{2} (f(-\infty,t) - f(\infty,t))(x - ct)\right)}} \quad (15)$$

where the wave speed ( $c$ ) is: (16)

$$c = \frac{f(-\infty,t) + f(\infty,t)}{2}$$

The dependent variable asymptotically approaches the boundary limits  $f(-\infty,t)$  to the left and  $f(\infty,t)$  to the right. By selecting the parameters  $f(-\infty,t) = 1$  and  $f(\infty,t) = -1$ , the wave speed  $c$  is set to zero. This simplifies the solution and eliminates the time dependency.

To approximate these limits, the boundary conditions for the numerical method were set at:  $f(-\infty,t) \approx f(-1,t) = 1$  and  $f(\infty,t) \approx f(1,t) = -1$ . This shortens the spatial domain to

[-1,1]. Although these boundary conditions are only an approximation, the error they introduce is significantly less than the computational precision for any Reynolds number greater than 40. Given these boundaries, the solution to Burger's equation simplifies to:

$$f(x) = \frac{1 - e^{(Re x)}}{1 + e^{(Re x)}} \quad (17)$$

To solve equation (14), fifty total points were distributed over the domain [-1,1]. The solution used a fourth order Taylor series with four cloud points. The Reynolds number was 50.

Figure 8 shows the residual and average error during the calculations. Again, the residual consistently decreased. The average error, however, reached a minimum after 5962 iterations and then increased back to a steady-state error of 0.003.

In Figure 9, the numerical solution of equation (14) is compared to the analytical exact solution. The exact and numerical derivative solutions are compared in Figure 10. To improve visualization, only the domain between -0.5 and 0.5 is shown in the figures.

Although this equation is considerably more difficult to model numerically, the gridless technique has provided an adequate solution. As expected, the primary errors occur in the

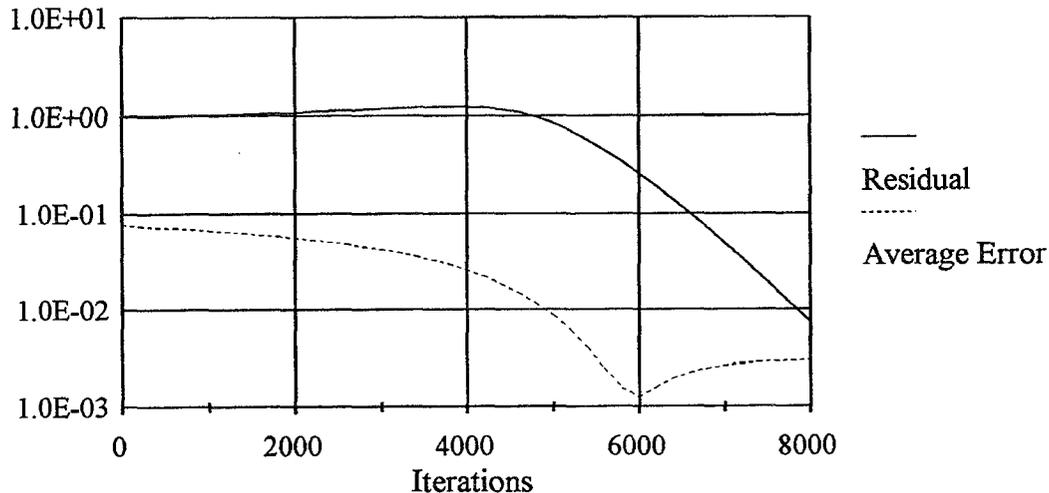


Figure 8. Residual and average error behavior for equation (14).

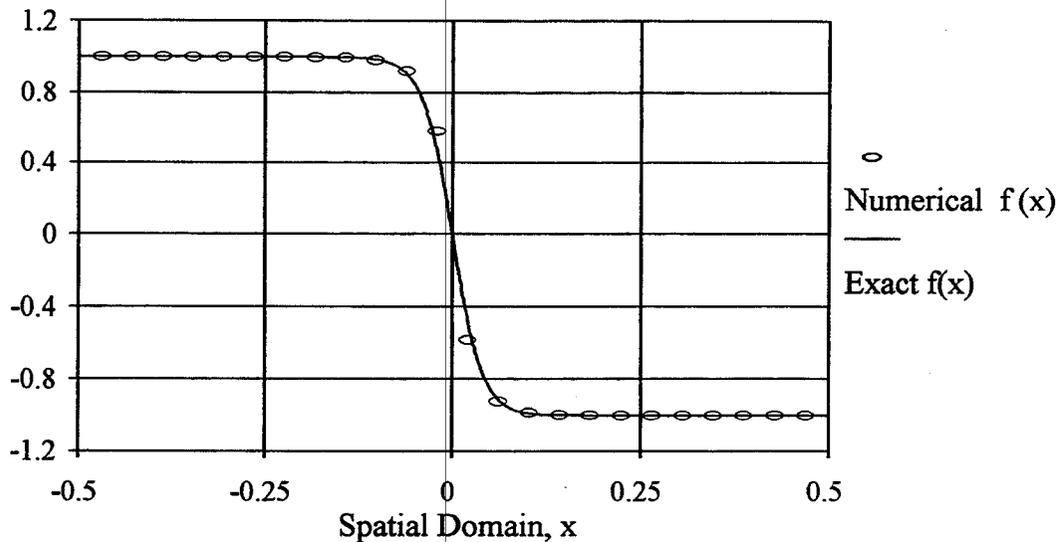


Figure 9. Solution to equation (14).

region of the shock. This results from the large gradient in this area and not having a sufficient number of points in this region to accurately model the behavior. With the introduction of adaption, this solution should improve.

### Summary

To solve each of the test equations from this chapter, the method required the Taylor series order, the number of neighboring cloud points, and the total number of grid points. Each of these values is discussed further in Chapter 3. In addition, this chapter defined the average error (E), the residual (R), and the convergence parameter ( $\epsilon$ ). Each of these definitions will be used substantially throughout this thesis. The convergence parameter ( $\epsilon$ ) determines when the method has achieved a steady state solution. Typical values range from 0.0001 to 0.001.

In general, the gridless technique without adaption effectively and efficiently solved each of the test equations. The solution to equation (10) demonstrates the method may provide significant accuracy under the correct conditions, while the solution to equation

(14) shows the gridless method's capability to model complex nonlinear behavior. Introducing adaption into the gridless methodology should only improve these results.

## CHAPTER 3

### DEVELOPMENT OF SUGGESTED DEFAULT PARAMETERS

#### Introduction

As mentioned briefly in Chapters 1 and 2, the gridless method must know the number of neighboring cloud points to utilize as well as the order of the Taylor series approximation. Although these two numbers are independent of each other, the number of cloud points must be greater than or equal to the order of the Taylor series. This chapter investigates the number of neighboring cloud points to use under most conditions, the order for the Taylor series approximation ( $p$ -adaption), and the effect the total number of points has on accuracy ( $h$ -adaption).

To achieve a level of semi-autonomy in the gridless program as desired in the first adaption goal, suggested default values were developed. The Fortran code used these values for all subsequent studies. To determine the default values, several test cases were considered.

First, a second-order Taylor series was used to solve each equation. The number of cloud points was varied from two through ten. Next, the order of the Taylor series was increased to three, and the range of cloud points varied from three through ten. The matrix of information from these tests ranged from a second-order approximation with only two cloud points through a tenth-order approximation with ten neighboring cloud points. For each test, the total number of iterations, the final residual, and the final average error were recorded.

Table 1 lists the information developed from these calculations using equations (10) and (12) as the governing equation. Each solution used twenty-five equally spaced points while the Taylor series order and number of neighbors varied. Computations ceased after the solution converged satisfying equation (9) for the user-defined convergence tolerance ( $\epsilon$ ).

## The Number of Neighboring Cloud Points

The minimum average error for each Taylor series order has been highlighted in Table 1. In nearly every case, the smallest error in the function values occurred using the minimum number of neighboring points. In other words, the best solution occurred when the number of neighboring cloud points equaled the order of the approximating Taylor series. This combination also usually converged in the minimum number of iterations.

This result is not unexpected. As more points are included in the cloud, these additional points will be farther from the focus point. Therefore, the information they provide about the derivatives at the focus point is less accurate than information from closer points.

Also, when the number of cloud points equals the order of the Taylor series approximation, a  $n \times n$   $[A]$  matrix forms. By eliminating the least-squares approximation necessary when the number of cloud points exceeds the order of the approximating series, the linear system of equations  $[A][Df] = [f]$  becomes better conditioned. This better conditioned system will have less error introduced by numerical inefficiency, and should be more accurate.

## The Taylor Series Approximation Order (p-adaption)

Adaption by higher order approximations, or p-adaption, has the capability to significantly increase accuracy in the numerical model. As expected, this effect on the gridless methodology seems to follow similar trends found in more traditional techniques. Table 1 and Figure 11 verify this behavior. Although p-adaption could improve accuracy while solving a transient problem, it will not substantially benefit the gridless method in modeling a moving boundary. Therefore, the Taylor series order will be fixed during all computations.

To determine the best Taylor series approximation order, two primary variables must be considered—average error and computation time. The behavior of the average error of the function can be determined from Table 1. However, the effects on computation time are somewhat more arbitrary.

As the order of the approximating Taylor series increases, the size of the  $[A]$  matrix also increases. This will substantially increase the number of calculations required to solve the system  $[A][Df] = [f]$ , and therefore increase the total computation time.

Considering only multiplication steps, approximately one-third  $n^3$  calculations are required to perform Gaussian elimination with partial pivoting on the  $n \times n$   $[A]$  matrix. However, this must only be performed each iteration the point distribution adapts. To perform this type of elimination on the vector  $[f]$  of dimension  $n$  requires approximately one-half  $n^2$  calculations, and the subsequent back-substitution to obtain the solution requires approximately an additional one-half  $n^2$  calculations. Since both of these steps must be performed every iteration, we will assume at least  $n^2$  calculations for each iteration.

Since  $n$  represents the Taylor series order, we will assume that the computational time increases exponentially with the approximating order. From Table 1, it is observed that the average error decreases approximately exponentially. The approximate optimum balance between these two variables can be visually determined from Figure 11. The figure represents increase in computational time and decrease in the final average function error. All values are normalized to fall between zero and one. From this information, a Taylor series order of four seems most appropriate.

Another important variable to consider before determining the order of the Taylor series approximation for any individual problem is the order of the governing equation to be solved. From the numerical investigation, it was found that the results from the highest order terms are often slightly in error. Therefore, the best results will be achieved if the order of the Taylor series approximation is at least one order, and preferably two orders, higher than the order of the equation being solved by the gridless method.

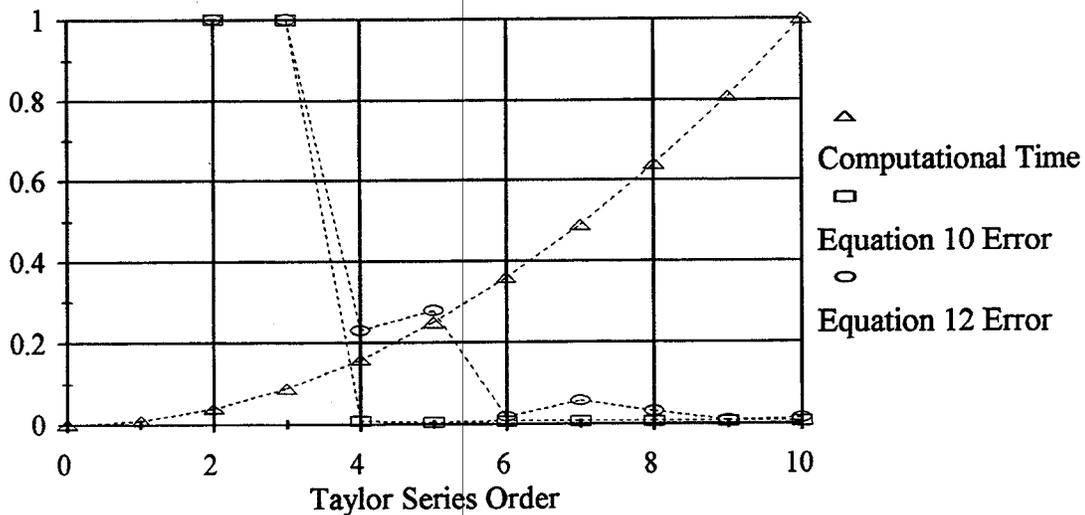


Figure 11. Balancing computation time and error.

Table 1.--Error Matrix for the Taylor Series Order and Cloud Point Number

Order	Neighbors	Equation 10		Equation 12	
		Iterations	Final Error	Iterations	Final Error
2	2	2696	9.963086E-05	150	0.0050541146
2	3	2711	0.0005420659	1363	0.0311862092
2	4	2696	0.0001889641	1657	0.0435034356
2	5	2662	0.0011128834	528	0.0586987167
2	6	2564	0.0036526151	2028	0.2369301705
3	3	4040	9.962296E-05	221	0.0050541065
3	4	4054	0.0003827897	1802	0.0202995014
3	5	4057	0.0004334706	2188	0.0273109506
3	6	4077	0.0008051579	2610	0.0682541894
3	7	4082	0.0009116358	2813	0.1095290158
3	8	4104	0.0013265712	3159	0.2611466792
3	9	4109	0.0014337887	3342	0.4268128423
3	10	4124	0.0017440142	3607	0.8287935343
4	4	5377	8.322905E-07	1656	0.0011659711
4	5	5376	5.450998E-06	2788	0.0145550826
4	6	5377	2.01865E-06	932	0.0020772534
4	7	5377	3.990729E-06	3578	0.0680901838
4	8	5376	4.833197E-06	2905	0.0192232538
4	9	5376	8.052892E-06	3516	0.0655569626
4	10	5376	1.11369E-05	3489	0.0666653416
5	5	6719	4.864279E-07	2174	0.0014083162
5	6	6719	1.078412E-06	2944	0.0054547002
5	7	6719	1.454839E-06	3224	0.0086365139
5	8	6719	5.105592E-06	3872	0.0260756623
5	9	6720	6.323026E-06	4059	0.0357441999
5	10	6720	1.275306E-05	4395	0.0635335325
6	6	8061	6.993621E-07	868	9.025581E-05
6	7	8061	6.909738E-07	1944	0.0010326111
6	8	8061	6.982564E-07	414	0.0003343229
6	9	8061	7.017185E-07	3134	0.0031099879
6	10	8061	6.90472E-07	1806	0.0020727045
7	7	9403	7.043219E-07	1778	0.0002924461
7	8	9403	6.967641E-07	1711	0.0003434345
7	9	9403	6.943673E-07	2275	0.0005890673
7	10	9403	6.683455E-07	3760	0.0034287514
8	8	10746	7.054241E-07	1545	0.00015738
8	9	10746	7.055134E-07	2722	0.0005988618
8	10	10746	7.055669E-07	2946	0.0007967186
9	9	12088	7.062041E-07	769	4.443921E-05
9	10	12086	7.075724E-07	1494	0.0001107909
10	10	13430	7.068726E-07	1208	7.12771E-05

### The Number of Mesh Points (h-adaption)

The effects of point spacing on accuracy were studied through several numerical experiments in which the number of grid points varied. The results of these tests for equations (10), (12), and (14) are given in Table 2.

Table 2.--Error Matrix for the Total Number of Mesh Points

Points	Equation 10	Equation 12	Equation 14
10	4.8568E-05	0.23228	no solution
20	1.4378E-06	0.0023532	no solution
30	6.7309E-07	0.00051469	no solution
40	5.6226E-07	0.00010794	no solution
50	5.0399E-07	2.2687E-05	0.0031278
60	4.6196E-07	8.5952E-06	0.0010626
70	4.289E-07	3.8529E-06	0.00049317
80	4.021E-07	1.9271E-06	0.00026651
90	3.7965E-07	1.0474E-06	0.00015753
100	3.6057E-07	6.0719E-07	9.8956E-05

As expected, decreasing point spacing has a substantial influence on the ultimate accuracy. Figure 12 demonstrates this result for equation (12).

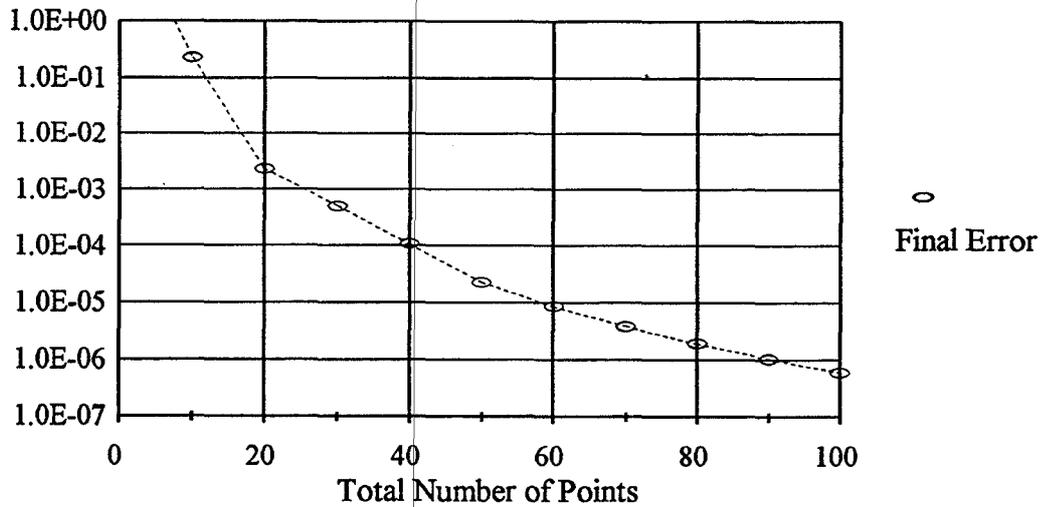


Figure 12. Effect of the number of points on final error for equation (12).

These tests were conducted only to verify that the gridless method follows trends similar to those of traditional techniques with regard to point spacing. Although adaption by mesh refinement will be introduced into the adaptive gridless code, these effects are not investigated further. It is sufficient to conclude that increasing the number of equally spaced points may significantly increase accuracy.

### Summary

The best accuracy for any given Taylor series order occurred when the number of cloud points equaled the approximation order. As the Taylor series order increased, the computation time required also increased substantially. To balance accuracy and computation time, a fourth-order Taylor series using four neighboring cloud points is suggested.

Although increasing the Taylor series order or decreasing point spacing substantially increases solution accuracy, improved accuracy does not improve the ability to correctly adapt the point distribution for a time-dependent problem. This requires actually moving point locations within the spatial domain.

## CHAPTER 4

### PROBLEM CONDITIONING

#### The Matrix Condition Number

When referring to numerical systems, the condition of a problem indicates the solution's sensitivity to small changes in the data. If the solution is not unreasonably sensitive to the data, the problem is well conditioned. Conversely, a problem is ill conditioned if small relative changes in the data produce a large change in the answers. The condition number of a matrix measures this sensitivity. A large condition number indicates an ill-conditioned matrix, and the solution should be accepted with some scepticism.

For any invertible matrix  $[A]$  the condition number ( $\kappa$ ) is

$$\kappa = \|A\| \|A^{-1}\| \quad (18)$$

where  $\|A\|$  represents the norm of the matrix  $[A]$ . Therefore, the condition number ( $\kappa$ ) depends on the type of norm used in the calculation. However, since the condition number's order of magnitude is of primary interest, any norm will obtain similar results.

For the system of equations  $[A][x] = [b]$ , the condition number of  $[A]$  indicates the expected relative error in the solution  $[x]$  is no greater than  $\kappa$  times the relative error in the input information  $[b]$  (Kincaid and Cheney 1991). Considering the order of magnitude of  $\kappa$ ,  $b$ , and  $x$  to be  $\theta$ ,  $\beta$ , and  $\phi$ , respectively, we may conclude:

$$(\text{relative } x \text{ error}) \leq (\kappa)(\text{relative } b \text{ error}) \quad (19)$$

$$10^\phi \leq 10^\theta 10^\beta$$

$$\phi \leq \theta + \beta$$

$$\text{digits of accuracy} \geq -(\theta + \beta)$$

Since  $\theta$  gives the order of magnitude of the condition number and  $\beta$  generally represents the precision of the machine being used, we may expect that the solution  $[x]$  will have at least  $-(\theta + \beta)$  digits of accuracy.

For convenience, the system of equations solved by the gridless method is listed again as:

$$[A][Df] = [f]$$

$$[A] = \begin{bmatrix} h_1 & \frac{h_1^2}{2!} & \frac{h_1^3}{3!} & \dots & \frac{h_1^n}{n!} \\ h_2 & \frac{h_2^2}{2!} & \frac{h_2^3}{3!} & \dots & \frac{h_2^n}{n!} \\ h_3 & \frac{h_3^2}{2!} & \frac{h_3^3}{3!} & \dots & \frac{h_3^n}{n!} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_n & \frac{h_n^2}{2!} & \frac{h_n^3}{3!} & \dots & \frac{h_n^n}{n!} \end{bmatrix} \quad [Df] = \begin{bmatrix} \frac{df_0}{dx} \\ \frac{d^2f_0}{dx^2} \\ \frac{d^3f_0}{dx^3} \\ \vdots \\ \frac{d^nf_0}{dx^n} \end{bmatrix} \quad [f] = \begin{bmatrix} (f_1 - f_0) \\ (f_2 - f_0) \\ (f_3 - f_0) \\ \vdots \\ (f_n - f_0) \end{bmatrix} \quad (3)$$

$$\text{where } h_i = (x_i - x_0)$$

In terms of these variables, the condition number of  $[A]$  measures the amplification of the error in  $[f]$  into the solution  $[Df]$ . Since the condition number is calculated only from the information contained within the matrix  $[A]$ , it is independent of the governing equation, the function values and derivatives, and the solution method (i.e., Gaussian elimination). However,  $[A]$  depends on the order of the Taylor series ( $n$ ) and the grid spacing ( $h$ ). When the matrix  $[A]$  is over determined, the number of neighbors will also affect the condition number.

To determine the effect of each of these three variables on the condition number of the system, several tests were conducted. In these tests, the number of points varied from 10 to 100, the Taylor series order varied from two through ten, and the number of neighboring cloud points varied from the current Taylor series order through ten. To keep the tests as uniform as possible, the condition number was calculated at a point equidistant from each of the boundaries. For example, with ten total points in the domain, the condition number was found at point 5; if the domain contained fifty total points, the condition number was found at point 25. The points were uniformly distributed across the domain  $[0,1]$  for each test. Therefore, adding more points acted as a measure of the point spacing ( $h$ ). For points near the boundaries, the condition number tended to increase slightly. However, it never

increased more than one order of magnitude. Table 3 lists a simplified form of the results from these tests.

The condition number for this study was computed by the subroutine *dgeco* found in the LINPACK library. This subroutine actually returns a reciprocal condition number ( $1/\kappa$ ). Although the method used to calculate this condition number was not investigated, several comparisons were made with the method previously described. The order of magnitude for each method was always the same.

### Number of Neighboring Cloud Points

Figure 13 depicts the effects of the number of points included in the cloud of neighbors. Although this graph utilizes the condition numbers obtained using a fourth-order Taylor series and fifty total points, all graphs of other combinations are very similar. The best condition number for each case tested occurred when the number of cloud points equaled the number of the Taylor series order. For this case, the  $[A]$  matrix had dimensions of  $nxn$ , and a least-squares approximation was not required. For the over determined cases, the matrix condition improved as more neighbors were included.

Although the condition number was substantially higher for the over determined systems, these numbers are completely dependent on the method used to solve the least-squares problem. Other solution techniques for an over determined system may provide improved conditioning.

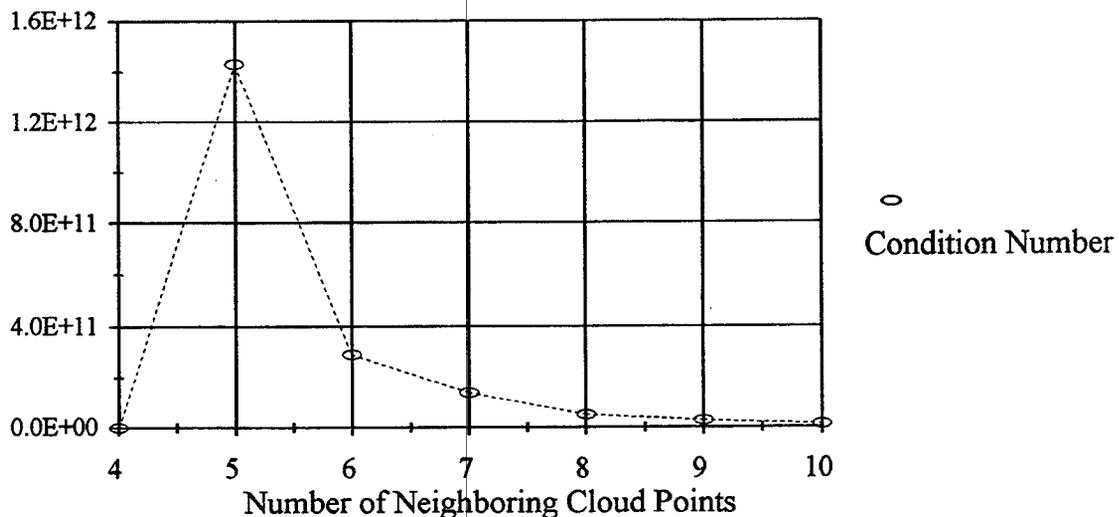


Figure 13. Condition number vs. number of cloud points.

Table 3.--Condition Numbers

Points	Order	Neighbors	Condition	Points	Order	Neighbors	Condition
20	2	2	38	60	6	6	1.02E+11
20	2	4	424	60	6	8	1.82E+20
20	2	6	205	60	6	10	1.53E+19
20	2	8	121	60	8	8	2.13E+15
20	2	10	80.1	60	8	9	3.43E+29
20	4	4	140000	60	8	10	3.72E+28
20	4	6	988000000	60	9	9	3.14E+17
20	4	8	170000000	60	9	10	5.77E+33
20	4	10	45300000	60	10	10	4.08E+19
20	6	6	354000000	80	2	2	158
20	6	8	2.19E+15	80	2	4	7340
20	6	10	1.85E+14	80	2	6	3570
20	8	8	7.67E+11	80	2	8	2110
20	8	10	4.84E+21	80	2	10	1400
20	10	10	1.52E+15	80	4	4	10100000
40	2	2	78	80	4	6	5.09E+12
40	2	4	1790	80	4	8	8.74E+11
40	2	6	868	80	4	10	2.33E+11
40	2	8	515	80	6	6	4.4E+11
40	2	10	341	80	6	8	3.37E+21
40	4	4	1210000	80	6	10	2.83E+20
40	4	6	7.38E+10	80	8	8	1.65E+16
40	4	8	1.27E+10	80	8	10	2.22E+30
40	4	10	3.37E+09	80	10	10	5.65E+20
40	6	6	1.29E+10	100	2	2	198
40	6	8	2.9E+18	100	2	4	11500
40	6	10	2.44E+17	100	2	6	5600
40	8	8	1.18E+14	100	2	8	3320
40	8	10	1.13E+26	100	2	10	2200
40	10	10	9.84E+17	100	4	4	19800000
60	2	2	118	100	4	6	1.97E+13
60	2	4	4090	100	4	8	3.39E+12
60	2	6	1990	100	4	10	9.02E+11
60	2	8	1180	100	6	6	1.36E+12
60	2	10	781	100	6	8	3.22E+22
60	4	4	4190000	100	6	10	2.71E+21
60	4	6	8.84E+11	100	8	8	7.99E+16
60	4	8	1.52E+11	100	8	10	5.22E+31
60	4	10	4.04E+10	100	10	10	4.3E+21

## Total Number of Points (Spacing)

Results of this study show that as the spacing is decreased, the range of values from the left to the right side of  $[A]$  will increase exponentially. This large difference in values causes  $[A]^{-1}$  to be large. Subsequently, the condition number will be large. Therefore, as the spacing decreases (more points are added), the condition number should increase. These conclusions are verified by Figure 14.

This represents the data for a fourth-order Taylor series using four cloud points. Graphs for other variations are very similar. As expected, the condition number increases as the spacing decreases. However, the total increase is only a couple of orders in magnitude. Therefore, only one or two digits of accuracy will be lost in the solution across the entire range of spacing tested.

Based on this study, point location (spacing) and the condition number of the local coefficient matrix are related. However, the dependent relationship is weak and does not provide any strong insight into adaption methodology. Consequently, dependence on condition number was not included in adaption criteria in this study.

## Taylor Series Order

In matrix  $[A]$ , as the Taylor series order ( $n$ ) increases the terms to the right become smaller. As the difference in the terms on the left and right of the matrix increases, the magnitude of the terms in the inverse matrix  $[A]^{-1}$  becomes very large. This will cause the condition number to become large. Because the factorial of  $n$  magnifies the effect of the Taylor series order on the condition number, this is the limiting variable on the condition number's magnitude. Figure 15 demonstrates this result. Note the log scale in the figure for the y-axis.

Although this graph represents a test conducted with fifty total points, and the number of cloud points was always equal to the order of the Taylor series order, similar graphs may be obtained from any combination.

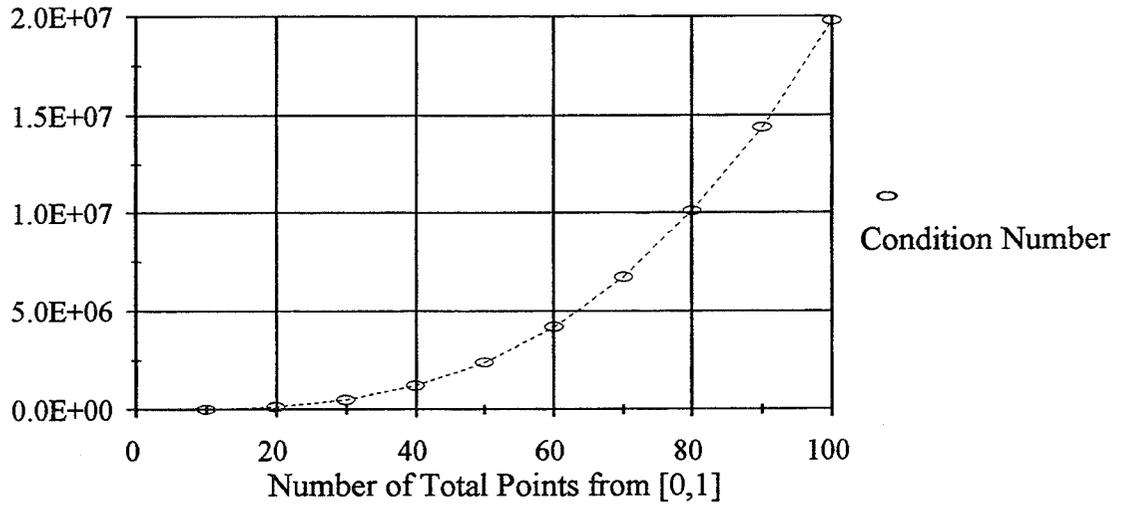


Figure 14. Condition number vs. number of points.

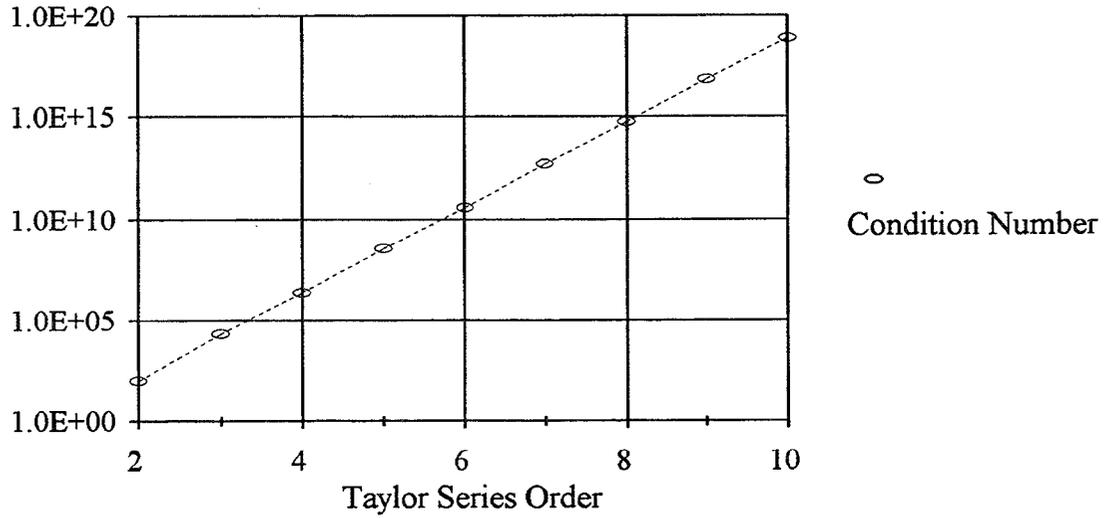


Figure 15. Condition number vs. Taylor series order.

## Guaranteed Digits of Accuracy

As previously explained, at least  $-(\theta + \beta)$  digits of accuracy can be expected in the derivative terms.  $\theta$  represents the order of the condition number, and  $\beta$  represents the error in the input information or machine precision. Since the Fortran code was written in double precision, machine precision was  $10^{-16}$  ( $\beta = 16$ ). Therefore, at least  $(16 - \theta)$  digits of accuracy may be expected.

Using this equation, a fourth-order Taylor series, four cloud points, and the information in Table 3, at least ten through twelve digits of accuracy are guaranteed for each spacing tested. Again, the point spacing does not have a considerable effect. However, changing the order of the Taylor series has a significant effect on the accuracy that is guaranteed. Figure 16 represents the effects of changing the Taylor series order on the guaranteed accuracy. This graph keeps the number of cloud points equal to the Taylor series order and uses fifty points.

## Summary

As the Taylor series order increases, the guaranteed accuracy in the solution decreases from poor conditioning of the numerical system. Although in many cases zero digits of accuracy are guaranteed, several digits of accuracy may still be achieved. For example, in

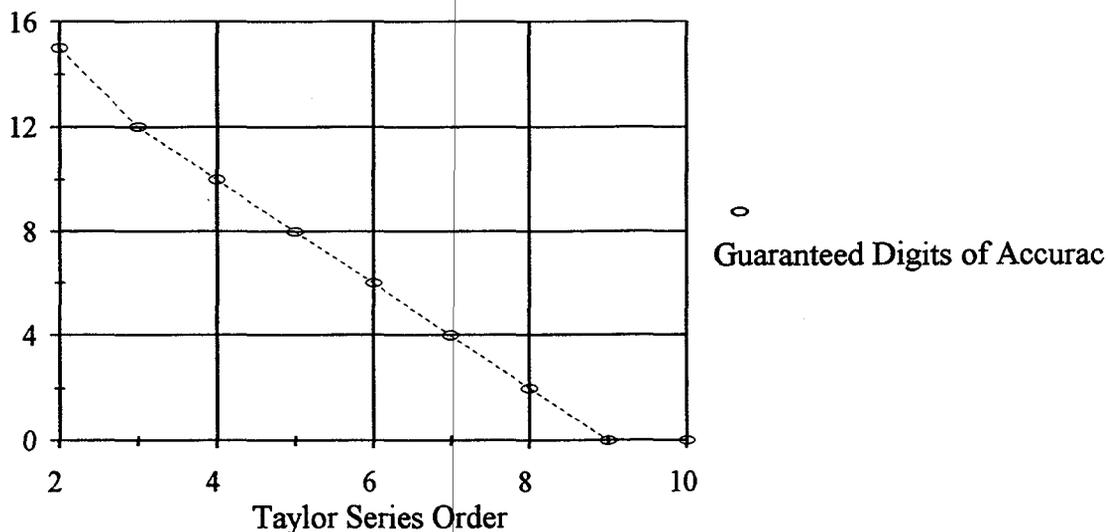


Figure 16. Guaranteed digits of accuracy for a given Taylor series order.

approximations even though the condition number for these cases is extremely high. Obviously, these solutions still have a large degree of accuracy. However, this accuracy is not guaranteed, and the unstable system of equations may produce erroneous solutions without warning. To ensure stability in the equations being solved, and to guarantee some degree of accuracy, higher order Taylor series approximations should be avoided.

The relationship between “nearest neighbor” point location and condition number was examined. As expected, the condition number increases as the spacing decreases. However, the total increase is only a couple orders of magnitude over the entire range of spacing tested. Therefore, the relationship between point location and condition number does not appear valuable for defining adaption criteria.

## CHAPTER 5

### ADAPTION BY POINT MOVEMENT

#### When to Begin Adaption

Prior to engaging any form of adaption, the process must evaluate the need for adaption. Generally, the adaption process introduces additional calculations and temporarily increases the error. However, adaption should ultimately provide a more accurate solution with a minimum number of additional calculations. To support these goals, adaption in any form is only initiated under two conditions. First, if the process has progressed to a nearly converged solution, adaption is introduced to refine this solution; or second, adaption is introduced as a last action for recovering a rapidly diverging solution.

When a solution begins to diverge, the residual will rapidly increase, often by an order of magnitude with each iteration. Adaption begins if the residual for the current iteration is 100 times the initial residual. If the solution is diverging from a mesh-related problem, adaption may have the capability to recover the solution. Once the solution again begins to converge, adaption discontinues.

Adaption begins when the solution is nearly converged. This occurs when the product of ten times the convergence parameter times the current time increment has the same magnitude as the summation of the residual over ten iterations.

$$10 \Delta t \epsilon > \sum_{i=k-10}^k R_i \quad (20)$$

*i is the iteration number*  
*k is the current iteration*

By not adapting until the solution is nearly converged, the process adapts to a more correct solution and unnecessary calculations are eliminated. Once this adaption begins, it continues until the governing equation is solved, even if the residual change increases above  $10 \Delta t \epsilon$ .

The choice of an appropriate value for the convergence parameter ( $\epsilon$ ) is left to the program user. This value will strongly affect the total computational time and ultimate

accuracy. A suggested default value is 0.001. For the examples tested, this value offered sufficient convergence and a near minimum error.

### **Adaption by Point Movement**

Once the solution has converged sufficiently and if adaption is necessary, the first type of adaption implemented is point movement. The technique utilized in the Fortran code for moving points differs slightly from previous research (Wolfe et al. 1996).

Local adaption by point movement utilizes a spring analogy approach. In this technique, the mesh structure moves toward an equilibrium distribution in which a local spring coefficient times the distance between points remains constant. This is analogous to an object placed between two physical springs moving to a position where the forces acting on it from each spring sum to zero. Just as the spring constant reflects the actual stiffness of a physical spring, the spring coefficient in the numerical system reflects the stiffness, or resistance to movement, of each point in the mesh.

For an actual spring, physical laws dictate the properties that determine the spring's stiffness. For the theoretical spring system, however, the stiffness may be defined. The properties used to define the stiffness should emphasize the important characteristics of the function's behavior. Therefore, the adaption algorithm utilizes the dependent variable, the gradient, and the second derivative when calculating the numerical stiffness. This departs slightly from adaption in many commercial solvers, which only adapt on the gradient of the dependent variable. However, allowing differences in both the variable magnitude and its first and second derivatives to affect the numerical stiffness both generalizes and moderates the method.

To determine when adaption is necessary, two spring coefficients are calculated for each point. The mathematical construction of the numerical stiffness coefficients is given in equation (21). These coefficients essentially represent a linear sum of the changes that occur in the function and its first and second derivatives between the central point  $I$  and its neighbors ( $I-1$  or  $I+1$ ). Each of the three components is uniformly weighted. To keep the magnitude of each component similar, the change in the values is divided by the total range.

$$k_{i-1} = \left( \left| \frac{f(x)_i - f(x)_{i-1}}{(f(x)_{\max} - f(x)_{\min})} \right| + \left| \frac{f'(x)_i - f'(x)_{i-1}}{(f'(x)_{\max} - f'(x)_{\min})} \right| + \left| \frac{f''(x)_i - f''(x)_{i-1}}{(f''(x)_{\max} - f''(x)_{\min})} \right| \right) \quad (21)$$

$$k_{i+1} = \left( \left| \frac{f(x)_i - f(x)_{i+1}}{(f(x)_{\max} - f(x)_{\min})} \right| + \left| \frac{f'(x)_i - f'(x)_{i+1}}{(f'(x)_{\max} - f'(x)_{\min})} \right| + \left| \frac{f''(x)_i - f''(x)_{i+1}}{(f''(x)_{\max} - f''(x)_{\min})} \right| \right)$$

These adaption stiffness coefficients are utilized to form the generalized forces given in equation (22). The term  $F_{i-1-i}$  refers to the generalized force acting on point  $I$  by point  $I-1$ . The adaption algorithm will continue to move the mesh points until these two forces are approximately equal.

$$F_{i-1-i} = k_{i-1} \cdot (x_i - x_{i-1}) \quad (22)$$

$$F_{i+1-i} = k_{i+1} \cdot (x_{i+1} - x_i)$$

If the spring coefficients ( $k$ ) did not change with the relative positions of the points, the distance ( $\Delta x$ ) point  $I$  must move to make equation (21) equal would be:

$$k_{i-1} \cdot ((x_i + \Delta x) - x_{i-1}) = k_{i+1} \cdot (x_{i+1} - (x_i + \Delta x)) \quad (22)$$

$$\Delta x = \frac{k_{i-1} \cdot (x_{i-1} - x_i) + k_{i+1} \cdot (x_{i+1} - x_i)}{k_{i-1} + k_{i+1}}$$

To adjust for the change this movement will cause on the spring coefficient ( $k$ ), the point is actually only moved a fraction of this distance. The distance the point  $I$  moves in the numerical model is given by equation (23). By moving only a fraction of the distance, the movement algorithm progresses slower but is much more stable.

$$\Delta x = \frac{k_{i-1} \cdot (x_{i-1} - x_i) + k_{i+1} \cdot (x_{i+1} - x_i)}{20 (k_{i-1} + k_{i+1})} \quad (23)$$

### When Adaption is Necessary

A user-defined movement parameter controls when adaption by point movement is necessary. This movement parameter ( $\alpha$ ) represents the amount of change allowed in the generalized forces from the left to the right boundaries. This type of approach has the advantage of giving a physical significance to the movement parameter. However, as more points are added to the system, the same movement parameter will actually allow less tolerance in the generalized force from one point to the next. If the movement parameter had been defined as the allowable change in the generalized forces between consecutive points, the total allowable change in the generalized forces could become larger than the actual total change between boundaries as points are added. Since this would disable point movement adaption, the movement parameter is defined as the total allowable change.

The allowable tolerance between the generalized forces is directly proportional to the movement parameter. This relationship is developed as follows. The point movement algorithm continues until the absolute difference between the generalized forces of equation (21) decrease below a predefined tolerance.

$$|F_{i-1-i} - F_{i+1-i}| < tolerance \quad (24)$$

This tolerance represents the amount of change possible in the generalized forces between two consecutive points. Since the movement parameter ( $\alpha$ ) is the maximum allowable change in the generalized forces from the left to the right boundaries, the relationship between the tolerance and the movement parameter is:

$$\alpha = tolerance \cdot (number\ of\ points - 1) \quad (25)$$

Therefore, adaption must continue until equation (26) is satisfied for every point in the domain.

$$|F_{i-1-i} - F_{i+1-i}| < \frac{\alpha}{number\ of\ points - 1} \quad (26)$$

## Selecting an Appropriate Point Movement Parameter

To determine an appropriate movement parameter ( $\alpha$ ), the values of each of the components of the generalized forces must be considered. Assume that when adaption by point movement is complete, the difference in the generalized forces is approximately the allowable tolerance.

$$\begin{aligned} |F_{i-1-i} - F_{i+1-i}| &\approx \textit{tolerance} \\ |k_{i-1} \cdot (x_i - x_{i-1}) - k_{i+1} \cdot (x_{i+1} - x_i)| &\approx \textit{tolerance} \end{aligned} \quad (27)$$

Physical significance may be given to the movement parameter if the approximate magnitude of each of the components in the generalized forces is known. By dividing the stiffness ( $k$ ) by the maximum stiffness between any two points across the domain ( $k_{\max}$ ), a normalized value between 0 and 1 results. Since  $x$  ranges from  $x_{\min}$  to  $x_{\max}$  (defined by the boundary conditions), the average distance between points ( $\Delta x_{\text{ave}}$ ) is:

$$\Delta x_{\text{ave}} = \frac{x_{\max} - x_{\min}}{\textit{number of points} - 1} \quad (28)$$

The maximum tolerance could be found using the maximum distance between two consecutive points. However, since the stiffness ( $k$ ) is generally less than one, the tolerance between the generalized forces can be estimated by using the average distance between points ( $\Delta x_{\text{ave}}$ ) and setting  $k$  equal to one.

$$\begin{aligned} \frac{\textit{maximum tolerance}}{k_{\max}} &\approx \left( \frac{k}{k_{\max}} \cdot \Delta x_{\text{ave}} \right) \\ \frac{\textit{maximum tolerance}}{k_{\max}} &\approx (1 \cdot \Delta x_{\text{ave}}) \\ \textit{maximum tolerance} &\approx k_{\max} \cdot \frac{x_{\max} - x_{\min}}{\textit{number of points} - 1} \end{aligned} \quad (29)$$

By inserting the movement parameter ( $\alpha$ ) into equation (29), the maximum change in the generalized forces across the domain is determined. As expected, this simplifies to the range of  $x$ .

$$\frac{\text{maximum } \alpha}{\text{number of points} - 1} \approx k_{\max} \cdot \frac{x_{\max} - x_{\min}}{\text{number of points} - 1} \quad (30)$$

$$\text{maximum } \alpha \approx k_{\max} \cdot (x_{\max} - x_{\min})$$

At this point, the range of the movement parameter ( $\alpha$ ) may be defined to vary from 0 to 1 by simply dividing by the maximum value ( $k_{\max} \cdot (x_{\max} - x_{\min})$ ). Therefore, the movement parameter will range from 0 to 1, and this division will occur automatically within the program.

Since the maximum  $\alpha$  represents a 100% change in the generalized forces, this value would require very little, if any, adaption. Therefore, the movement parameter should be a small fraction of this maximum. Lower values will require additional iterations, but they will also provide a more accurate simulation of the spring system. Typically a value of 0.10, or 10% of the maximum change, provides sufficient adaption.

### Point Movement Algorithm Performance

To enhance the capability of the numerical method, adaption should provide a more accurate solution. A more accurate numerical solution will globally approach the true solution and capture the important local features of the true solution. Adaption by point movement primarily positions points appropriately to capture the important features such as maximum and minimum values, or areas of steep gradient. By correctly positioning the mesh points, the average error should decrease. To verify the effectiveness of the point movement method, variations of the test cases from Chapter 2 were modeled.

All tests used the same boundary conditions, Taylor series order (fourth), number of cloud points (four), and number of total points as their respective tests cases from Chapter 2. In addition, the tests used a convergence parameter ( $\epsilon$ ) of 0.001 and a movement parameter ( $\alpha$ ) of 0.10. The simulations were not concluded when the convergence parameter was satisfied. Instead, calculations continued for a set "total time." This provided better comparisons with the previous examples. The convergence parameter was used, however, to initiate the adaption process.

### Point Movement Ability to Solve Equation (10) Regardless of the Initial Point Seeding

In recalling the results from solving equation (10) without adaption, the solution was smooth and contained no steep gradients. Without adaption for an initially evenly spaced grid, the average error was reduced to  $1.89 \times 10^{-7}$ . With these conditions, it is not surprising that, given the movement parameter of 0.10, the solution required no additional adaption.

However, since this equation is well behaved, it may assist in demonstrating a further capability of the point movement algorithm. Rather than using a uniformly spaced initial point distribution, the points were intentionally clustered. If working correctly, the point movement algorithm should be sufficiently robust to reposition these points to a more uniform spacing without losing any solution integrity.

The results of this test are displayed in Figure 17. The top line shows the initial clustering of the points. The middle line demonstrates the final point locations subsequent to repositioning by the movement algorithm. As observed, the point movement adaption has successfully repositioned the initially clustered points to a more uniform spacing. The bottom line represents an evenly spaced grid which required no adaption.

During the calculations, the initially clustered point distribution caused the solution to diverge. This initiated the point adaption algorithm, which then corrected the point distribution until the solution started converging. Figure 18 demonstrates the effect adaption had on the residual and error during calculations. Although several iterations were required to reposition the points, the subsequent graphs are nearly identical. The eventual minimum error produced after adaption was  $1.60 \times 10^{-7}$ , a slight reduction from the minimum error without adaption.

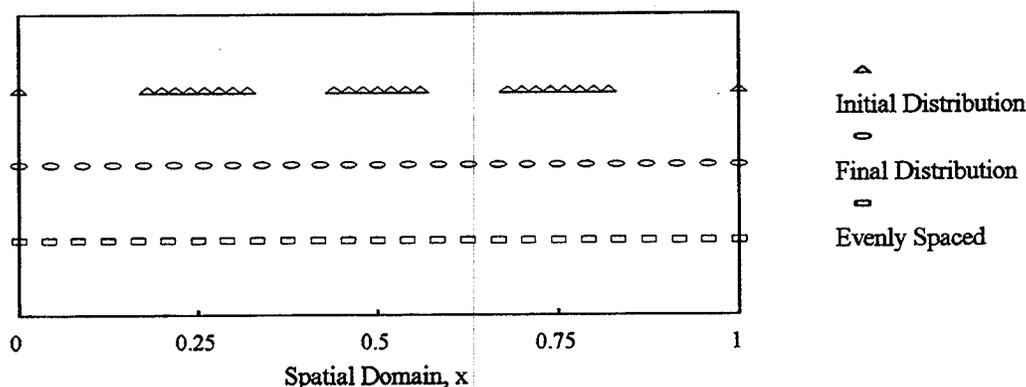


Figure 17. Repositioning of an initially clustered point distribution.

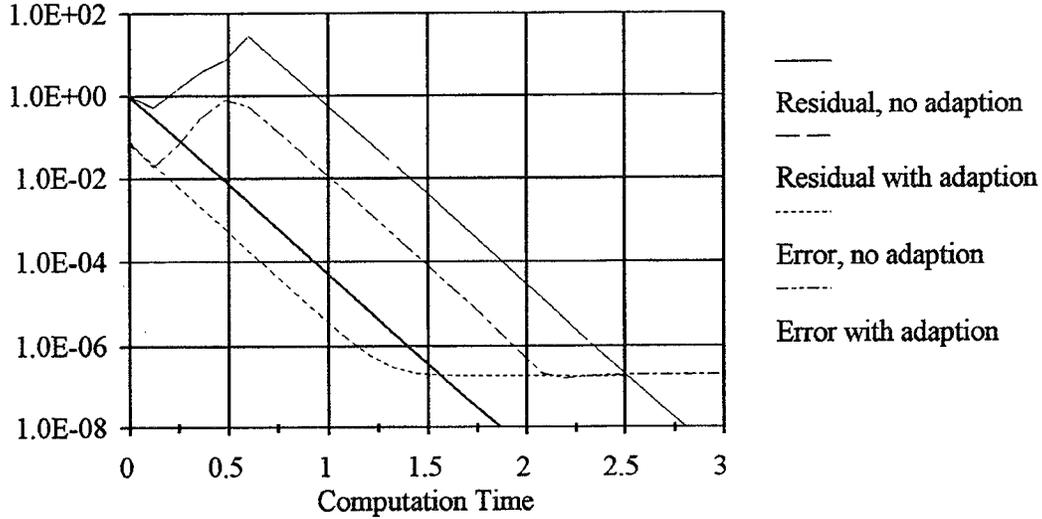


Figure 18. Residual and error comparisons with adaption for equation (10).

### Point Movement Adaption for Equation (14)

For a practical application of Burger's equation, the area in which the shock occurs is of the greatest interest. Although the numerical solution to Burger's equation without adaption found in Chapter 2 accurately represents the total solution across the domain, it does not sufficiently capture this area of steep gradient. Adaption by point movement helps to obtain the necessary clustering.

Figure 19 displays a solution to equation (14) applying adaption by point movement. Points have effectively been moved into the region of the shock. Although the numerical solution obtained with adaption is slightly offset to the left, it improves the solution obtained without adaption. With adaption by point movement applied, the average error changed very little. However, since more points are in the shock area, these points tended to have larger errors associated with them individually, and the maximum error was reduced.

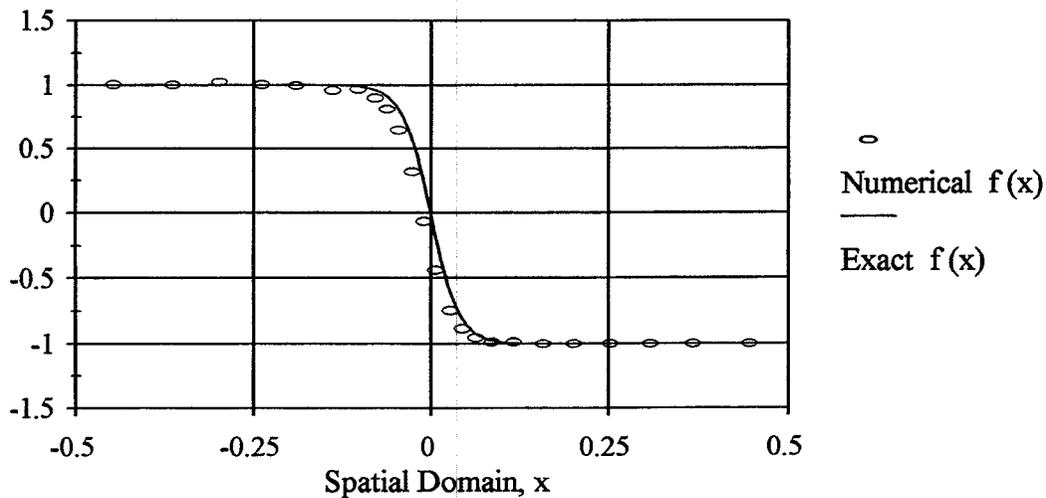


Figure 19. Solution to equation (14) with point movement adaption.

### Summary

The point movement algorithm follows a spring analogy. The numerical stiffness coefficient ( $k$ ) for this system is defined to be largest in the areas that require clustering. Generalized forces ( $F$ ) are generated by multiplying the stiffness coefficient times the distance between points. The adaption algorithm moves the points until these forces are approximately constant.

In addition to the convergence parameter ( $\epsilon$ ), adaption by point movement requires a new user-defined movement parameter ( $\alpha$ ). This parameter represents the percent change allowed in the generalized forces over the domain of the function. Lower values will require more adaption, but will better distribute points according to the spring analogy.

In general, the point movement algorithm seems stable and effective. The important local features of the function have been adequately captured. The method is sufficiently robust to achieve similar results regardless of the initial point distribution. Although point movement does not significantly improve the error, it does provide a better representation of the overall solution.

**Blank Page**

## CHAPTER 6

### THE ADAPTION COEFFICIENT

#### Introduction

As previously discussed, point movement adaption utilizes a spring-analogy approach. For a physical spring, the stiffness depends on the material properties. For the numerical system, however, the stiffness must be defined. The properties used to define the numerical stiffness should emphasize the important characteristics of the function's behavior. In addition, the stiffness must be defined in terms of the information known about the current solution and point distribution.

Typically, point clustering is desired in areas of steep gradients, near boundaries, or at maximum and minimum locations. These locations are indicated by the current approximation to the function value and the derivative terms. The current time and mesh spacing may also provide important information to use in developing the numerical stiffness. The adaption stiffness coefficient must utilize a combination of these values to provide the desired mesh distribution and improve accuracy.

This section provides a greater understanding of the adaption stiffness coefficient ( $k$ ) and its effect on point location and accuracy. No attempt has been made to find the optimal adaption coefficient. However, several possibilities are presented and discussed.

#### The Traditional Spring Analogy

The traditional approach places the spring between consecutive points. The adaption stiffness coefficient for this approach then compares the function value and derivatives (or whatever values are used in the stiffness definition) with the same values at neighboring points. The comparison may add, subtract, multiply, or divide these values. Multiplying the adaption stiffness coefficient by the distance between the two points obtains the generalized force. The point then moves to a location in which the generalized forces acting on it from each side are equal.

For the spring analogy method, the most logical formulation for the adaption stiffness coefficient is:

$$k_{i+1} = \left( c_0 \left| \frac{f_i - f_{i+1}}{f_{\max} - f_{\min}} \right| + c_1 \left| \frac{\frac{df}{dx}_i - \frac{df}{dx}_{i+1}}{\frac{df}{dx}_{\max} - \frac{df}{dx}_{\min}} \right| + \dots + c_n \left| \frac{\frac{df^n}{dx}_i - \frac{df^n}{dx}_{i+1}}{\frac{df^n}{dx}_{\max} - \frac{df^n}{dx}_{\min}} \right| \right) \quad (31)$$

where  $f$  represents  $f(x)$   
 $c_0, c_1, \dots, c_n$  are constants  
 $n$  represents the Taylor series order

This expression utilizes most of the information available for determining the stiffness between points  $i$  and  $i+1$ . The constants  $c_0$  through  $c_n$  may be varied to emphasize the various terms.

Equation (32) is a simplified form of equation (31). The coefficients  $c_0$ ,  $c_1$ , and  $c_2$  represent a percentage weighting on each term. They should add to 1.00 or 100%. To keep the magnitude of the adaption stiffness coefficient roughly equal to the stiffness coefficient used throughout this thesis, the whole equation is multiplied by three. Since equation (32) is similar to the adaption spring coefficient used for all other examples in this thesis, it will be studied in greater detail.

$$k_{i+1} = 3 \left( c_0 \left| \frac{f(x)_i - f(x)_{i+1}}{f(x)_{\max} - f(x)_{\min}} \right| + c_1 \left| \frac{f'(x)_i - f'(x)_{i+1}}{f'(x)_{\max} - f'(x)_{\min}} \right| + c_2 \left| \frac{f''(x)_i - f''(x)_{i+1}}{f''(x)_{\max} - f''(x)_{\min}} \right| \right) \quad (32)$$

$$c_0 + c_1 + c_2 = 1$$

To determine the effect of each term on the final accuracy, several tests were considered. For equations (10) and (12) in Chapter 2, point movement adaption was performed using equation (32) and a variety of percentage weightings for each term. The calculations for all three tests used a fourth-order Taylor series approximation with four cloud points, twenty-five initially equally spaced points, a movement parameter ( $\alpha$ ) of 0.1, and a convergence tolerance ( $\epsilon$ ) of 0.001.

First, equation (10) was solved using a range of weighting from zero to 100% for each term in equation (32). These results are given in Figure 20.

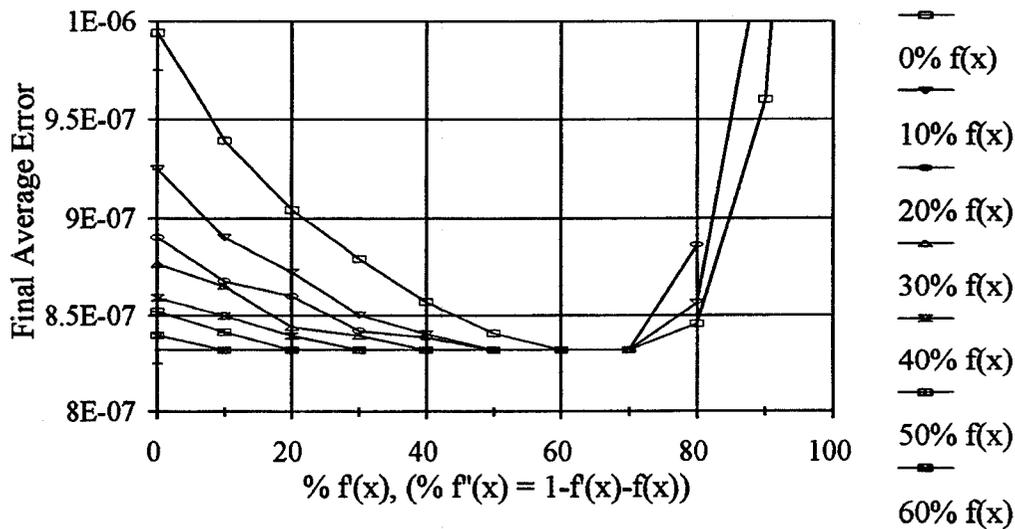


Figure 20. Final error for equation (10) with variable stiffness weightings.

According to this figure, the average error increases as the percent weighting on the function value ( $c_0$ ) increases. Also, a weighting greater than 40% and less than 70% is desired for the first derivative term ( $c_1$ ). A feasible weighting for the three terms is 40% on the function value, 40% on the first derivative, and 20% on the second derivative. Although this value does not provide the minimum error, its error is near the minimum and some moderation is maintained in the stiffness coefficient. Although it is not the minimum, an equal weighting on each term would provide a very good solution.

Figure 21 gives the results for testing equation (12). Again, the weighting on each term was varied from zero to 100%, and the final average error was recorded.

For equation (12), the percent weighting on the function value had very little effect on the final average error. However, the weighting on the first derivative term should remain between 20% and 80%. Although it did not provide the minimum error, an equal weighting on each of the three terms would have provided a very good solution.

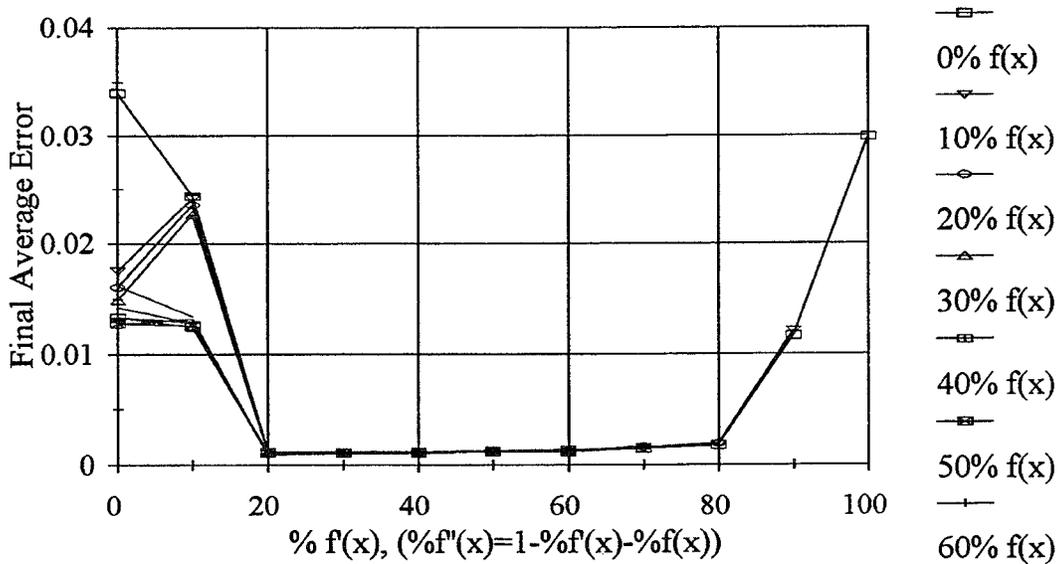


Figure 21. Final error for equation (12) with variable stiffness weightings.

### Summary

Developing an adaption stiffness coefficient ( $k$ ) that offers the desired movement of points in the spring analogy is challenging. The adaption stiffness coefficient must offer an appropriate clustering and improve accuracy. For local spring-analogy adaption methods, the stiffness coefficient generally compares known information about the function and derivatives with similar information at neighboring points. This is the method used for the point movement and point refinement algorithms in this report.

## CHAPTER 7

### ADAPTION BY MESH REFINEMENT

#### Introduction

To effectively model a moving boundary problem without losing accuracy, adaption by mesh refinement is required. As the problem domain expands, more points will be necessary. In contrast, points may be deleted if the domain becomes smaller. As seen in Table 3 of Chapter 3, adding points to the mesh may significantly reduce the final error in the solution. More points, however, increase the number of calculations for each iteration and usually requires additional iterations for the solution to converge. Adaption by mesh refinement attempts to balance these factors.

#### When Adaption by Refinement is Necessary

Adaption by mesh refinement utilizes the same spring analogy as point movement adaption. In general, if the average magnitude of the generalized forces from two consecutive points exceeds a user-defined refinement parameter ( $\gamma$ ), then a point is added in this interval. Conversely, if this generalized force for every point in the domain declines arbitrarily below three fourths the refinement parameter, a point is deleted from the system. Typical values for the refinement parameter range from 0.001 to 0.01.

*Add a point at  $i$  if:  $\gamma < F_{ave}$*

$$F_{ave} = \frac{F_{i-1} + F_{i+1}}{2} \quad (34)$$

*Delete a point if:  $\frac{3}{4}\gamma > F_{i-1}$*

*for every  $i = 2, 3, \dots$  number of points*

#### Sequence of Adaption Methods

Including mesh refinement with point movement complicates the adaption process. By using the spring analogy, the point movement algorithm adapts the mesh until the

generalized forces are approximately equal. Therefore, if one of the generalized forces exceeds the refinement parameter, theoretically the generalized forces between all the points could exceed the refinement parameter. This would require that a point be added between each interval, nearly doubling the total number of points. Subsequent adaptations may then require deleting many of these points. A more conservative method of limiting the addition and deletion of points must be followed.

In general, point movement should be nearly complete to accurately determine if points should be added or deleted. However, the effort to correctly position points by the movement algorithm may be wasted unless the total number of points is approximately correct before point movement begins. Once adaptation by mesh refinement begins, the method must add or delete points cautiously to maintain stability in the solution. To adapt most effectively and efficiently, the Fortran code proceeded with point movement and mesh refinement in three steps.

The first adaptation technique suggested is a relaxed form of the mesh refinement method. When the solution is nearly converged according to equation (20), this routine is initiated. Beginning at the left boundary and working to the right, the average generalized forces given by equation (34) are evaluated at each point. If the average generalized force for point  $i$  is greater than one and one half the refinement parameter, a new point is added next to point  $i$ . Since deleting points may cause a significant decrease in accuracy, no points are deleted in this relaxed mesh refinement.

$$A \text{ point is added at } i \text{ if: } \gamma < \frac{3}{2} F_{ave} \quad (35)$$

*No points are deleted*

If a point is added at  $i$ , then point  $i+1$  immediately to the right is not allowed to adapt during this iteration. Once the need for adaptation has been determined at each point, the program proceeds to the next iteration on the governing equation. This relaxed mesh refinement continues for every subsequent iteration until no points are added. After completing this initial relaxed mesh refinement adaptation, it is not used again.

The relaxed mesh refinement should obtain the approximate number of total mesh points needed to satisfy the refinement parameter. Next, adaptation by point movement proceeds.

The final adaptation step refines the mesh by adding or deleting points. When adaptation by point movement completes, all the generalized forces should be approximately equal. At

this point, the algorithm determines the need for further refinement. To eliminate the possibility of overrefining the mesh, only one point may be added or deleted each iteration. If the maximum average generalized force is greater than the refinement parameter, the adaption adds a single point at the location of the maximum value. Otherwise, if the minimum average generalized force declines below three fourths the refinement parameter, this point is deleted. If a point is added or deleted, the point movement algorithm must correctly reposition the points before the mesh refines further.

### **Adding Points**

The addition of a point using this adaption algorithm is a four-step process. First, adaption determines the distances to the nearest neighbors on either side of the central point. Second, it positions a new point one third the distance from the central point to the first neighbor on the right. Next, the central point moves one third the distance from its current position toward the nearest neighbor on the left. Finally, values of the dependent variable's magnitude and derivatives for the repositioned center point and the new point are derived using linear interpolation from the nearest neighboring points. This procedure attempts to maintain any clustering from previous adaptations, but allows the final positioning to be accomplished by the point movement adaption process.

### **Performance of the Combined Refinement and Movement Adaption Methods**

Equation (12), the second test case from Chapter 2, is appropriate to demonstrate the robustness of the combined refinement and movement methods. For the linear ODE, two sets of initial point distributions were utilized, and the results compared. The adaption algorithms should be sufficiently robust to determine the approximate optimum number of point for the solution and the given refinement parameter, regardless of the initial number of points. In the first test, the gridless calculations were initiated with 25 equally spaced points, while the second case utilized 100 points initially. For the first case, the refinement algorithm added points resulting in sixty-two total points in the final solution. The second test effectively reduced the number of points to a final total of sixty-five. Therefore, the refinement technique works very well in achieving similar point distribution for the solution regardless of the initial number of points for this particular ODE.

Next, the gridless method with full adaption capability solved the steady-state equations (10), (12), and (14). This verified the effectiveness of the adaption techniques to provide an accurate solution and capture the important features of the governing equation.

Each solution utilized a fourth-order Taylor series with four neighboring cloud points, a movement parameter of 0.1, and a convergence tolerance of 0.001. Equations (10) and (12) started with twenty-five initially equally spaced points with a refinement parameter of 0.01. Equation (14) used fifty initial points and a refinement parameter of 0.005.

As expected equation (10) solved easily with full adaption, as shown in Figure 22. During adaption the number of points reduced from twenty five to twenty four, and the final error increased to  $9.90 \times 10^{-7}$  with full adaption. Adaption also provided an improved solution for equation (12), as shown in Figure 23. In the final solution, the total number of points increased to 43 reducing the error to  $2.94 \times 10^{-4}$ .

The solution to equation (14) with full adaption also improved considerably, as shown in Figure 24. The region near the shock has been captured very accurately. However, some error has been propagated from the shock area into the smoother regions to the right and left. The final error for this solution was  $3.00 \times 10^{-3}$ .

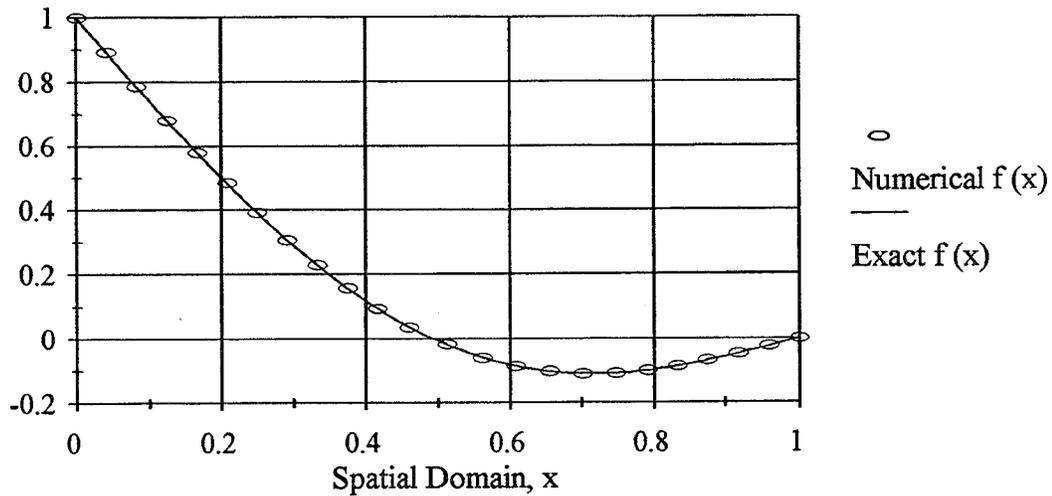


Figure 22. Solution to equation (10) with full adaption.

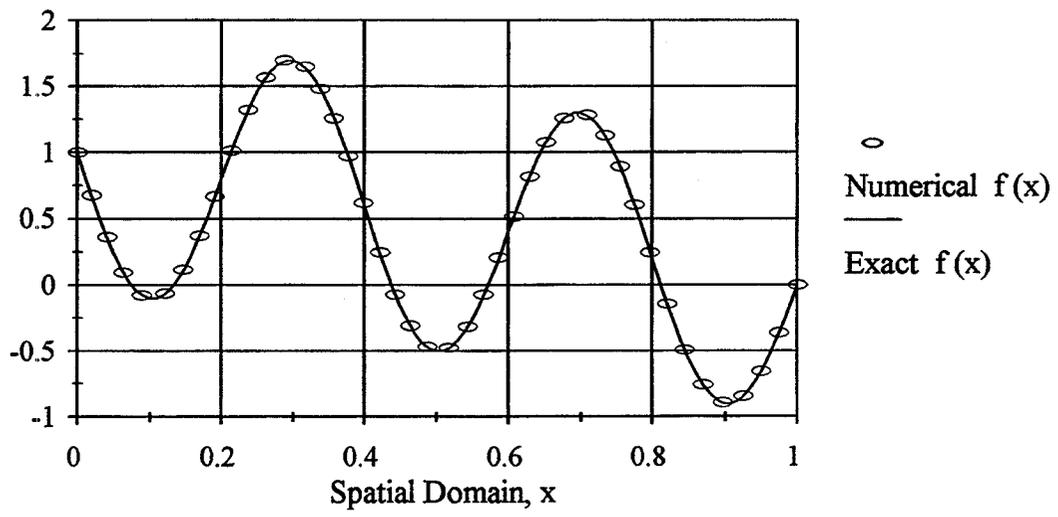


Figure 23. Solution to equation (12) with full adaption.

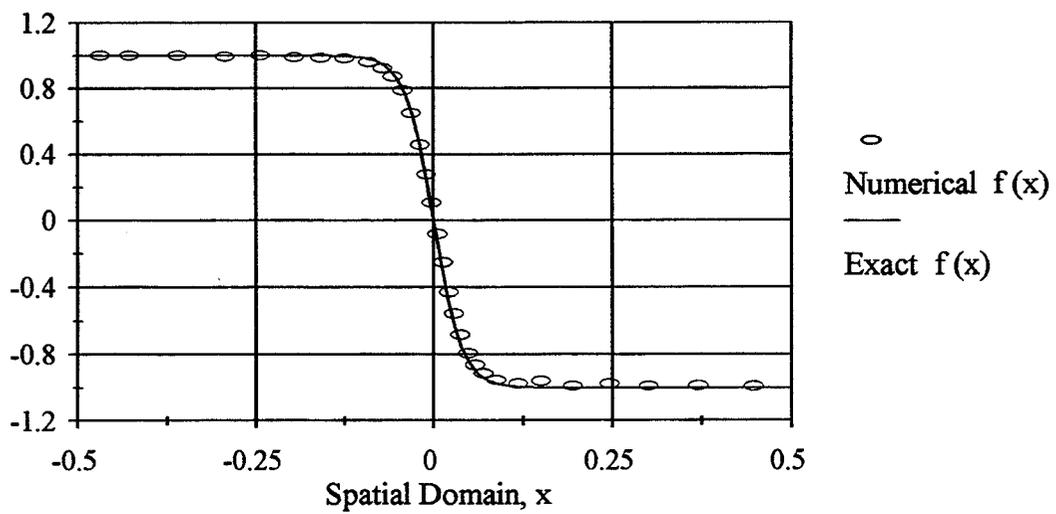


Figure 24. Solution to equation (14) with full adaption.

## Summary

Although adaption has introduced new restrictions and obstacles for the gridless methodology, it has improved the technique's ability to capture an accurate solution for these example. Generally, adaption increases the total computational time and does not provide a significant increase in the solution's average error. However, the method can effectively adapt while maintaining the solution stability and integrity. This fact verifies that the gridless method should be capable of modeling more complex transient problems such as a moving boundary.

## CHAPTER 8

### TRANSIENT SOLUTIONS USING THE GRIDLESS METHOD

#### Introduction

The objective of this research is to lay a foundation that will allow for accurate modeling of complex moving boundary problems. Traditional structured and unstructured solution methods are limited by moving boundaries. In addition to being computationally expensive, the constant remeshing during the transient calculations may cause the grid to distort and twist. By eliminating the permanent connectivity between points, gridless technology overcomes this obstacle.

Thus far, it has been demonstrated that the gridless method can solve a variety of linear and nonlinear steady-state problems, and adaption may be applied to improve these solutions. Next, the method is extended to allow for simple, time-varying problems. Eventually in this chapter, an actual moving boundary problem in one dimension will be modeled.

#### Extending the Solution Method for Solving Transient Problems

When solving the steady-state problems previously outlined, the numerical method started with a large error and large residual. Eventually the residual became very small and the solution was achieved.

Although the solution steps are the same, the approach for solving transient problems is slightly different. For transient problems the gridless method must maintain the solution in time, rather than to solve for an unknown solution. Since the initial function values are known, the initial error is zero. However, this error will immediately increase slightly to an approximately steady error. If the error continues to increase, a smaller time increment is most likely needed.

Solving a transient problem differs from solving a steady problem in three primary ways. First, computations generally proceed for a set length of time instead of stopping when the solution converges. Second, because the initial conditions are known, adaption may begin immediately. And third, the gridless method must constantly adapt since the actual solution is changing in time.

## A Simple Transient Test for the Gridless Method

The first transient problem modeled was the simple heat or diffusion equation with the diffusive constant set to one.

$$\frac{df(x)}{dt} = \frac{d^2f(x)}{dx^2} \quad (36)$$

For the boundary conditions  $f(0,t) = f(1,t) = 0$ , and the initial condition  $f(x,0) = x - x^2$ , a separation of variables approach to the analytical solution gives:

$$f(x,t) = -\frac{2}{\pi^3} \left[ \sum_{n=1}^{\infty} \left[ \frac{1}{n^3} (n\pi \sin(n\pi) + 2 \cos(n\pi) - 2) \sin(n\pi x) e^{-n^2 \pi^2 t} \right] \right] \quad (37)$$

For this example, the mesh initially had twenty five equally space points. A fourth-order Taylor series was used with four neighboring cloud points. The movement parameter was 0.2, and the refinement parameter was 0.02.

Figure 25 compares the numerical solution for equation (36) to the analytical solution from equation (37) at various times during calculations. The solid lines represent the

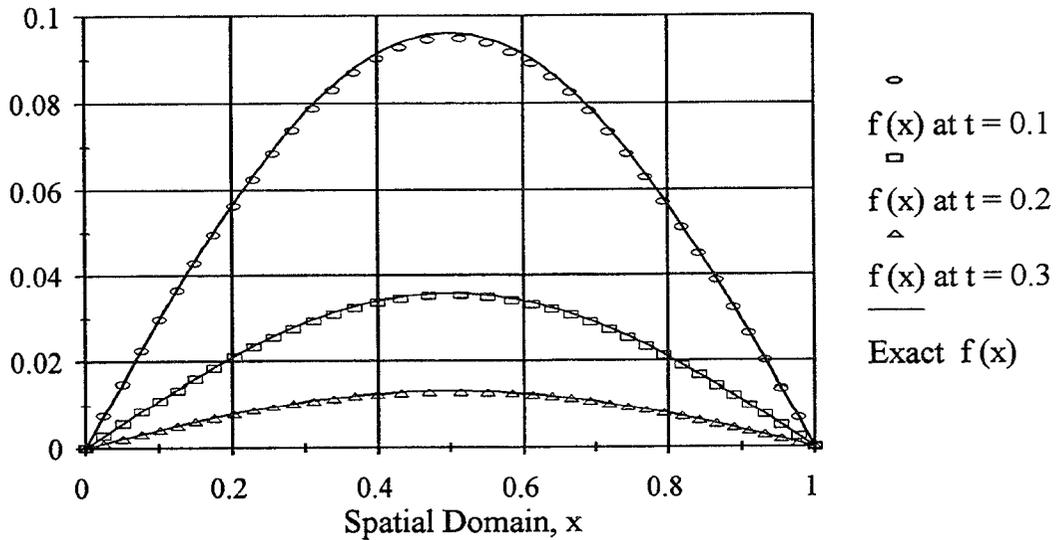


Figure 25. Solution to equation (36) at various times.

function's analytical solution for the respective times. As observed, the gridless method seems to track the solution very well for this simple linear partial differential equation.

### Numerical Modeling of a Transient Problem with a Time-Periodic Boundary Condition

Although the gridless method easily solved the previous example, the equation did not offer a substantial challenge for the method, and the problem eventually reduced to steady state with the solution  $f(x,t) = 0$ . The next example also involved solving the heat equation. However, a time-dependent condition was introduced on the left boundary. This is a true transient problem and will better test the gridless method's capabilities.

$$\frac{df(x)}{dt} = 0.01 \frac{d^2f(x)}{dx^2} \quad (38)$$

For the boundary and initial conditions of  $f(0,t) = \sin(10t)$ ,  $f(\infty,t) = 0$ , and  $f(x,0) = e^{(-x\sqrt{500})} \sin(-x\sqrt{500})$  the analytical solution to equation (37) is (Poulikakos, 1994):

$$f(x,t) = e^{(-x\sqrt{500})} \sin(10t - x\sqrt{500}) \quad (39)$$

Numerically, the boundary condition  $f(\infty, \cdot) = 0$  was estimated at  $f(\cdot, \cdot) = 0$ . This introduced an error on the order of  $10^{-10}$  on the right boundary. However, since this error was significantly less than the average error in the calculations of the gridless method, it was accepted.

Equation (38) was solved numerically using a fourth-order Taylor series with four neighboring cloud points. The movement parameter was 0.1, and the refinement parameter was 0.005. The initial mesh contained fifty equally spaced points. Results are shown in Figure 26.

Although this figure demonstrates the gridless method's ability to follow the motion of a moving solution in time, the method needs improvement. After 0.3 seconds, this solution contained twenty-one points with an average error of  $2.13 \times 10^{-3}$ . Especially in the region of the spatial domain from 0.2 through 0.4, the numerical and exact solutions differed noticeably.

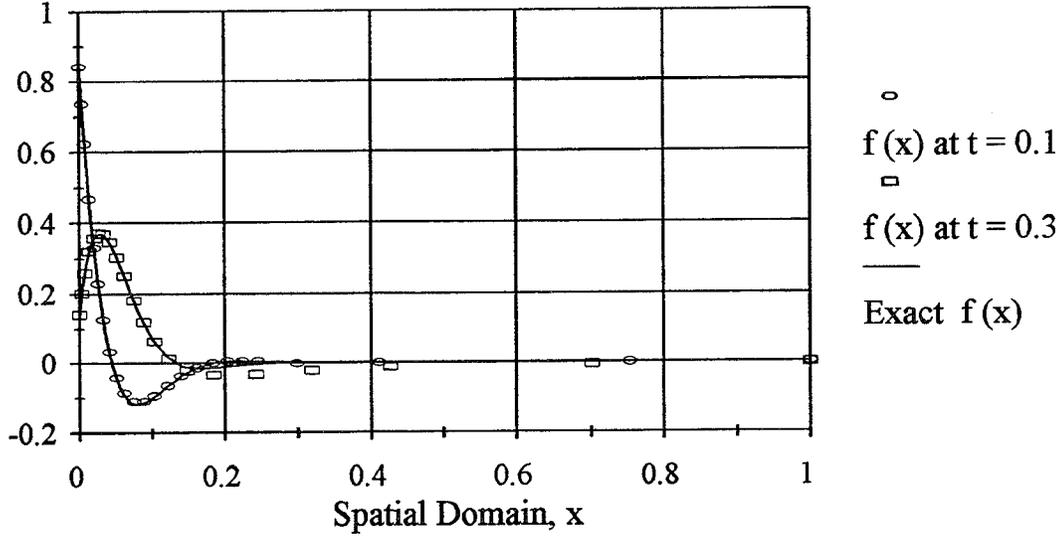


Figure 26. Solution to equation (38) at various times.

Perhaps the largest contributors to the errors for a time-accurate solution are the integration time increment and interpolation errors. Although the time step, according to equation (6), provides sufficient stability to model transient problems, accuracy should improve for smaller increments. For further examples, the time increment will be reduced arbitrarily by three fourths.

Adaption will always introduce error from interpolation, and adaption is required for solving moving boundary type problems. The optimum solution for these problems will balance the level of accuracy and the degree of adaption necessary. Relaxing the adaption techniques will limit the amount of adaption necessary. This requires fewer steps of adaption and fewer interpolations steps. Therefore, the induced error from adaption should decrease.

Relaxing the point movement parameter directly decreases the error. However, increasing the movement parameter will provide a less efficient point distribution, and the important function features may not be captured.

Typically, decreasing the refinement parameter should increase accuracy since this adds points. However, adding points requires interpolation over a large distance, introducing a significant error. These two factors nearly cancel each other. Therefore, the refinement parameter has little effect on accuracy for transient problems.

The refinement parameter has a secondary effect on accuracy. A lower refinement parameter will add points to the system. More points require additional adaption steps. Since more adaption steps introduce more error, a lower refinement parameter may actually increase the average error. If a tighter mesh is desired, this error must be accepted.

Generally, for steady-state problems the first adaption step is to add points using a relaxed refinement parameter to approximately obtain the correct number of points. Since adding points introduces large errors, this is inefficient for transient adaption methods. Instead, a good approximation to the number of points must be selected based in the initial conditions.

Another method for controlling the induced error is to limit the distance of the interpolation each time a point is moved. According to equation (23), each point may move 5% of the desired distance each iteration. By limiting this distance further, the induced error may be substantially reduced, but the number of adaption iteration required will increase. However, numerical experiments have shown that the cumulative effect of moving a shorter distance over more iterations reduces the error.

The final suggestion restricts the frequency of adaption. Requiring a few iterations between adaptations allows the gridless method time to recover from past errors before introducing new errors. This will moderate the method and reduce the error slightly.

In Figure 27, the solution to equation (38) is given using relaxed adaption. Again, the solution started with fifty evenly spaced points using a fourth-order Taylor series with four neighboring cloud points. The movement parameter was relaxed from 0.1 to 0.25, while the refinement parameter remained at 0.005. Points moved only 2% of the desired distance each iteration, and ten iterations were required between adaptations.

In Figure 28, the error for the first solution to equation (38), given by Figure 26, and the error for the second solution, given by Figure 27, are compared. For the second method, the error after 0.3 seconds has been reduced from  $2.13 \times 10^{-3}$  to  $6.991 \times 10^{-4}$ . Although the first solution is initially more accurate, the second solution is much more stable. This eventually leads to a more accurate solution.

### **A Moving Boundary Solution**

To verify the gridless method's capability to model an actual moving boundary, equation (36) was solved with a transient boundary condition. The temperature at each boundary was held at a constant zero value. However, the right boundary was allowed to

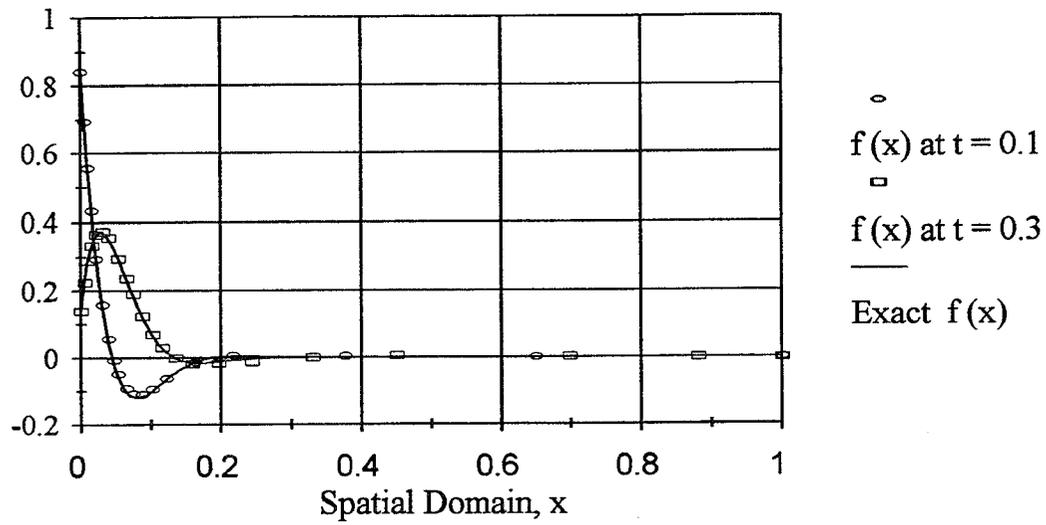


Figure 27. Solution to equation (38) at various times with relaxed adaption.

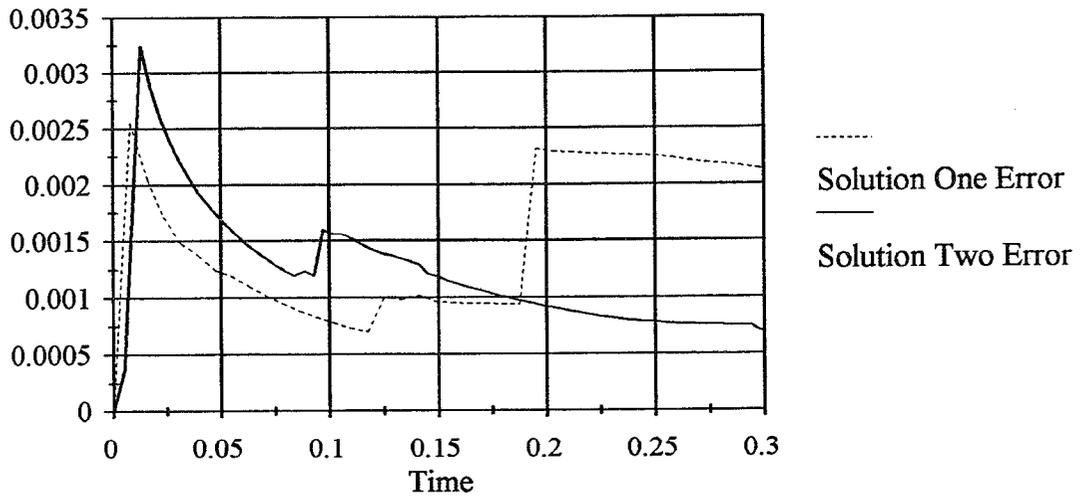


Figure 28. Error comparison for two solutions of equation (38).

move rapidly from 1.0 to 2.5 over the first 0.3 seconds of calculations. Although an analytical solution is not available for comparison, Figure 29 gives the numerical solution. The trend of the solution for the gridless method seems to be physically realistic.

### Grid Prediction

For transient problems, the adapted mesh tends to lag the true solution slightly. The adaption process attempts to fit the mesh to the current function solution. However, before the adaption is complete, the function has a new value and the point distribution is already incorrect. By predicting the expected function value and adapting toward this predicted value, the point distribution should become more time-accurate.

A simple mesh prediction method was introduced into the gridless program. This method utilized linear extrapolation to predict the function values at a future time. The function value at each point was stored for several iterations. No adaption could occur while storing these values since this would change the point locations, making the extrapolation invalid. After the new function values were predicted, the gridless program adapted as usual using the predicted values rather than the current function values.

To determine the effects of mesh prediction on the accuracy of the solution, equation (38) was solved using the prediction method. The linear extrapolation used the current function values and the function values from ten iterations previous. The function values were predicted twenty iterations into the future if the current time increment remained

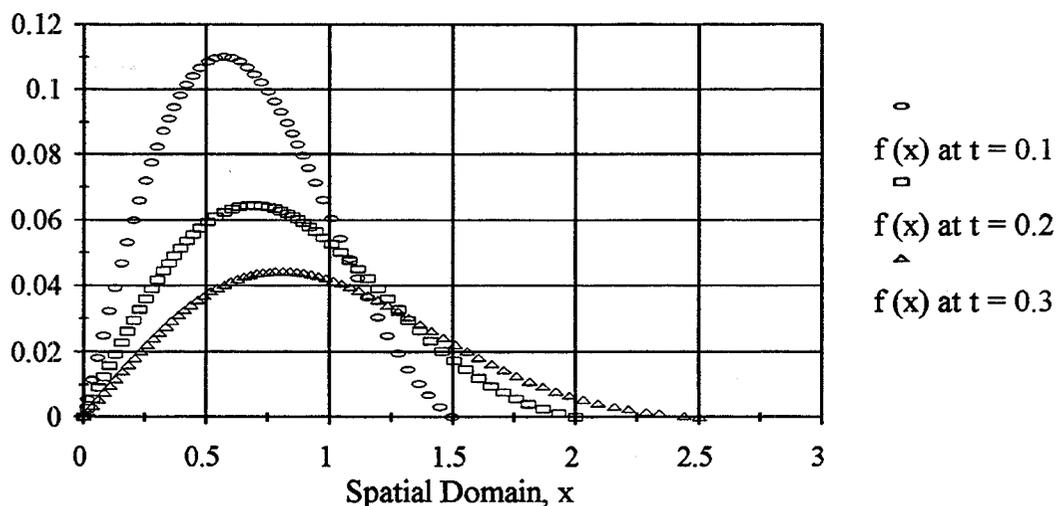


Figure 29. A moving boundary solution for equation (36).

constant. The error for this technique is compared with the error without prediction in Figure 30.

Although the point distribution may have been more accurate using the mesh prediction, the solution accuracy decreased. Since the predicted function values are only estimates, these values are not as stable as the actual function values for the current iteration. Therefore, the estimated values changed more than the current values, and they required more adaption. With more adaption required, the error increased.

### Summary

Transient problems offer an interesting challenge for the gridless method. Adaption inherently introduces error into the solution from linear interpolation on the new function values. Since the transient problem does not allow the gridless method sufficient time to recover from these induced errors, they must be kept as small as possible. This may be accomplished by interpolating over smaller distances, reducing the integration time increment, or simply not adapting as frequently. The adaption algorithm uses each of these ideas in some form to maintain time accuracy. Although a balance of accuracy, computation time, and the required level of adaption must be maintained, the gridless technique is capable of solving complex transient problems.

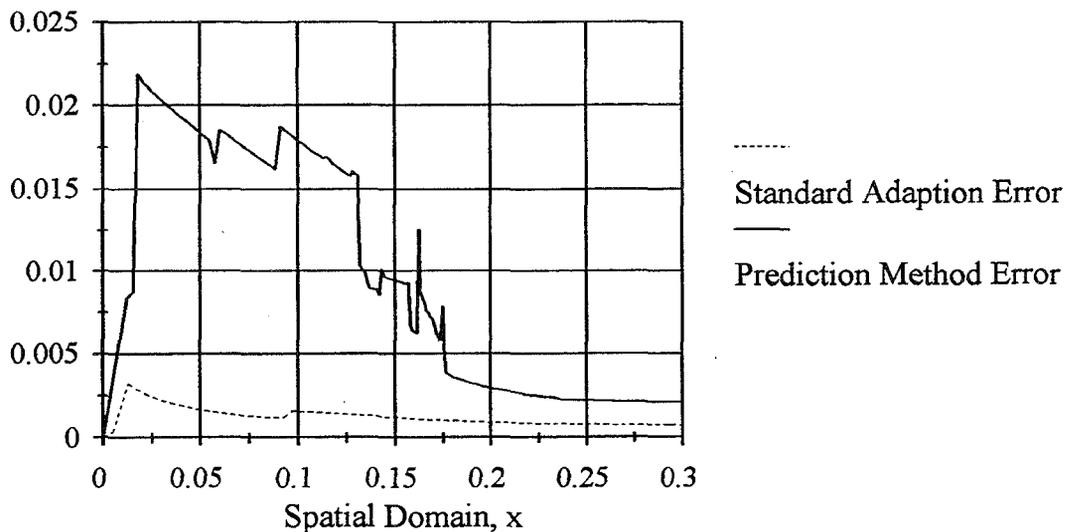


Figure 30. Error comparison for standard and prediction adaption methods.

## CHAPTER 9

### GLOBAL ADAPTION

#### Introduction to Global Adaption

Although adaption is necessary for the gridless method to model a complex transient problem such as a moving boundary, local point adaption introduces additional restrictions on the method. For steady problems, adaption increases computational time. For transient problems, constant local adaption introduces error into the solution each iteration as adaption occurs.

Local adaption compares the stiffness between neighboring points and adjusts points individually. These adjustments may then cause other neighboring points to require adjustment during subsequent iterations. During the iteration steps in which adaption occurs, the gridless method makes less progress toward the solution because of the errors induced by linear interpolation. Therefore, transient problems require smaller time increments to maintain accuracy, and steady problems need more iterations before they converge.

Global adaption compares the stiffness over the entire domain and then adapts accordingly. Rather than adapting points, this method generates a completely new mesh that perfectly matches the current function and stiffness values. Function values for the new mesh are found by linear interpolation between point locations from the previous mesh.

#### The Global Adaption Method

For global adaption, the adaption stiffness coefficient must be defined at each point as in equation (40) rather than between points. This definition includes information only about the function's first and second derivatives. Since information from the function's magnitude gives little information about an appropriate point distribution, this value is not included.

$$k_i = \left( \left| \frac{f'(x)_i}{f'(x)_{\max} - f'(x)_{\min}} \right| + \left| \frac{f''(x)_i}{f''(x)_{\max} - f''(x)_{\min}} \right| \right) \quad (40)$$

By using this definition of the adaption stiffness coefficient, the local generalized force is defined as the adaption coefficient ( $k$ ) times the length of the region the point occupies. Numerically integrating the adaption stiffness coefficient over the entire domain using the trapezoidal rule obtains the total generalized force. This is equivalent to finding the area under the stiffness curve.

$$F_{total} = \int k dx \quad (41)$$

$$F_{total} = \sum_{i=2}^{number\ of\ points} \frac{(k_i + k_{i-1})}{2} (x_i - x_{i-1})$$

For global adaption, a single global movement parameter ( $\alpha_{global}$ ) replaces both the point movement and point refinement parameters. This global movement parameter represents the maximum allowable value of the individual generalized force at any point.

$$\alpha_{global} = F_{i, max} \quad (42)$$

The new total number of points ( $n_{new}$ ) is determined by dividing the total generalized force by the global movement parameter and raising the result to the next higher integer.

$$n_{new} = integer \left( \frac{F_{total}}{\alpha} \right) + 1 \quad (43)$$

Dividing the total generalized force by the new total number of points gives the actual applied generalized force ( $F_{i, applied}$ ) for each point. This value will be slightly less than the global movement parameter.

$$F_{i, applied} = \frac{F_{total}}{n_{new}} \quad (44)$$

Equation (41) finds the approximate area under the stiffness curve. Again working with this equation from left to right, the total area is broken into ( $n_{new} - 1$ ) regions each with an area equal to  $F_{i, applied}$ . For the new mesh distribution, a point is placed where each of

these areas meets. The function values at the new point locations are found using linear interpolation between the grid points from the previous mesh.

The global adaption method provides a near perfect distribution in a single iteration. Only one iteration of accuracy is lost, and all subsequent iterations may proceed toward the correct solution without the need of constant adaption. However, during the one iteration for global adaption, nearly every point will move, causing a substantial increase in the error. The advantages and disadvantages between local and global adaption depend on the effects of these errors on the ultimate solution accuracy.

### **Global Adaption Performance for Steady-State Problems**

To determine the capabilities of the global adaption algorithm, it was used to solve problems previously solved with local adaption methods. Figure 31 gives the solution using global adaption for equation (12).

The error for the local and global techniques solving this equation is compared in Figure 32. The solution with local adaption finished in 3907 iterations with fifty nine total points and an average error of  $2.62 \times 10^{-4}$ . The global adaption technique adapting every 500 iterations required 3018 iterations and finished with an average error of  $2.87 \times 10^{-4}$ . The global movement parameter was adjusted to 0.0055 to achieve the same number of points in the final solution as the local adaption method. Although the error for each method is comparable, the global method converged considerably faster. In addition to requiring fewer iterations, the solution using the global method used fewer calculations each iteration.

To determine the effects on modeling transient problems, the gridless method with global adaption solved equation (36). Figure 33 gives this solution. Global adaption provided a very uniform point distribution for this problem.

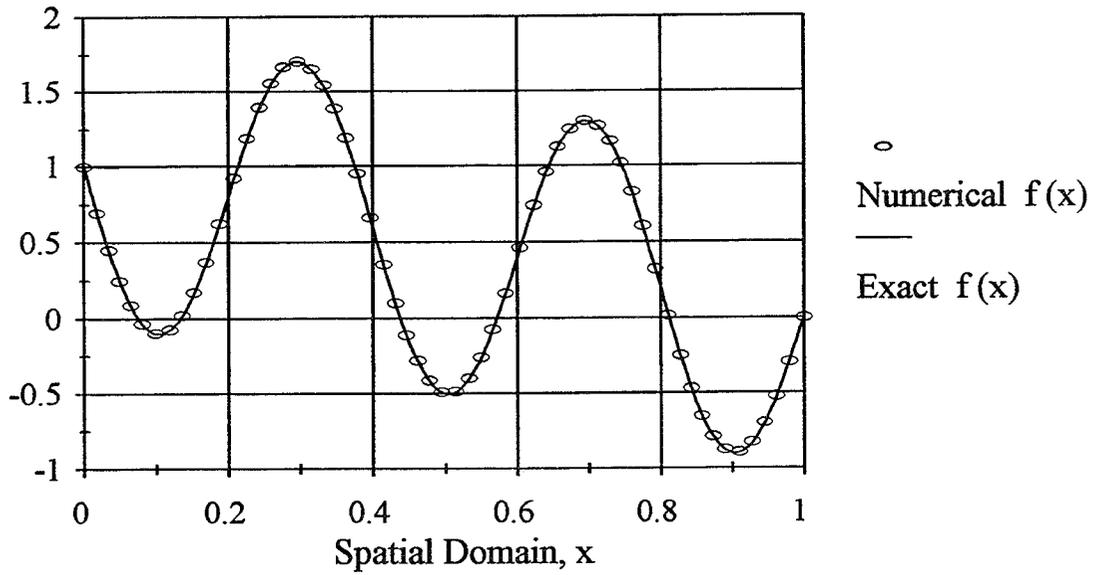


Figure 31. Solution to equation (12) with global adaption.

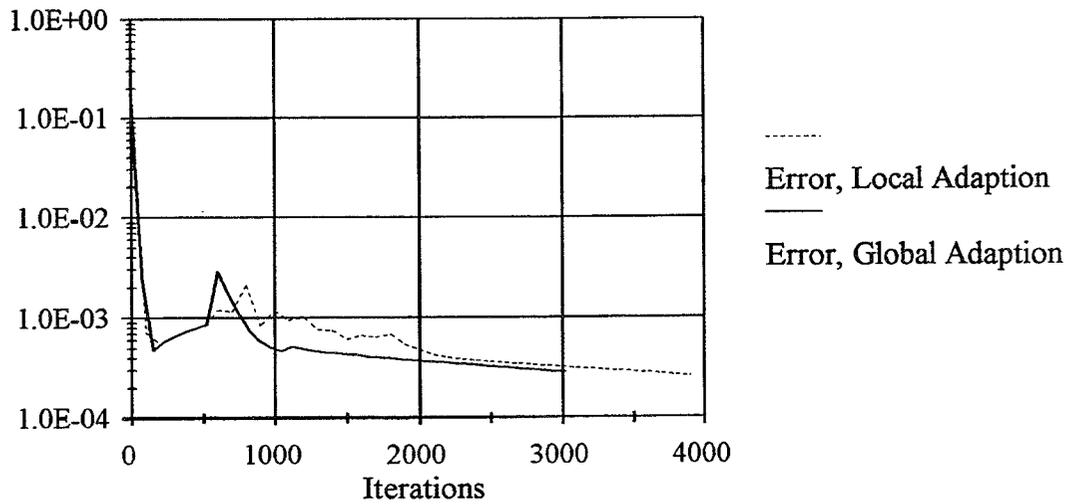


Figure 32. Error comparison for local and global solutions to equation (12).

Figure 34 compares the error for the local and global techniques solving equation (36). For local adaption, the problem required 17788 iterations to complete the solution over the first 0.3 seconds. It ended with twenty nine total points and an average error of  $3.12 \times 10^{-7}$ . The global adaption technique adapting every 500 iterations required 9882 iterations over the 0.3 seconds and finished with an average error of  $4.89 \times 10^{-6}$ . The global adaption coefficient was adjusted to 0.01675 to achieve the same number of points in the final solution as the local adaption method.

For transient solutions, the induced error from interpolation has a significant effect on the solution accuracy. Although global adaption requires fewer iterations and finishes the calculations much quicker, Figure 34 demonstrates that local adaption with consistently small induced errors better maintains accuracy in time for this problem.

### Summary

Local adaption methods require several iterations to complete, and error is introduced during each of these iterations. Global adaption provides a near perfect point distribution in one iteration. However, during this one iteration nearly every point will move, causing a substantial increase in the error.

For transient problems, the gridless technique cannot recover from this large error increase, and the overall problem accuracy decreases. However, for the steady-state example, similar accuracy using global adaption techniques was achieved for considerably fewer total iterations.

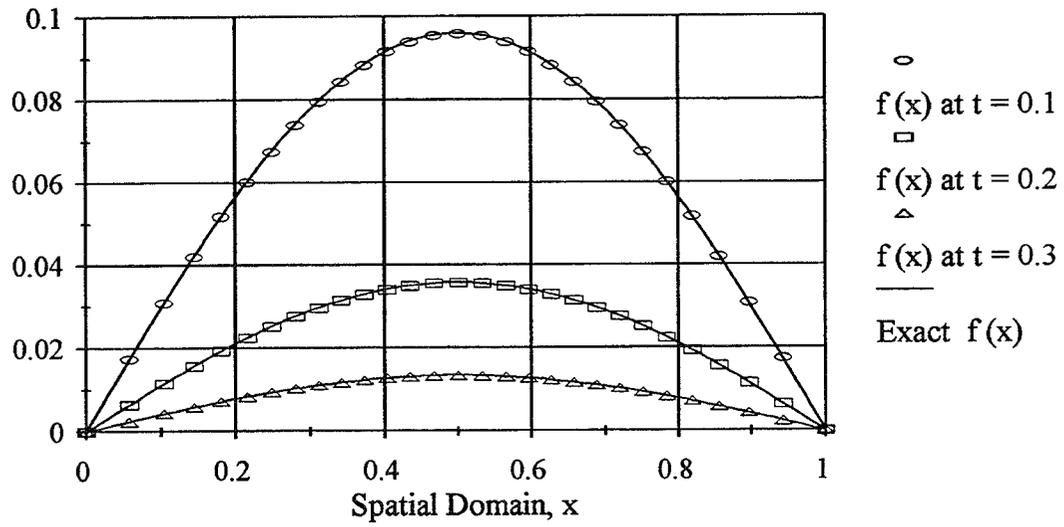


Figure 33. Solution to equation (36) with global adaption.

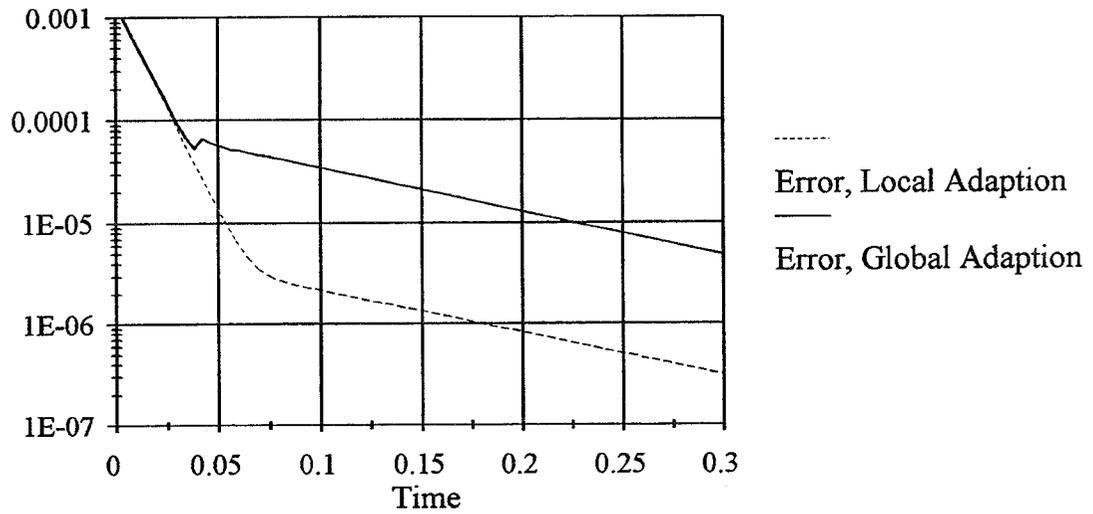


Figure 34. Error comparison for local and global solutions to equation (36).

## CHAPTER 10

### CONCLUSIONS

#### **The Strengths of the Gridless Method**

The strength of the gridless methodology lies in its ability to model any geometry using clouds of discrete points. Because these points are not required to maintain the consistent connectivity of grid-based solution methods, the point distribution cannot distort or twist into undesirable shapes as the mesh adapts. Therefore, gridless solutions maintain accuracy and integrity during adaption even in areas of high deformation or large flow gradients

For the one-dimensional problem, adaption must be applied to the gridless method to realize these capabilities. The two underlying objectives for this research were first, to verify the capabilities of the gridless method to adapt and provide accurate solutions for a variety of problem types, and second, to provide background information for implementing a multidimensional adaptive gridless method. The research centered on investigating a variety of problem types, mesh distributions, and adaption variation. No attempt was made to find an optimal one-dimensional gridless solution technique. Instead, several possibilities were studied that may apply to adaption in higher dimensions.

#### **Purpose of Adaption**

Adaption in any form should complement and enhance the numerical method. For adaption to be effective, it must capture the important features of the true solution and minimize error. Higher order approximations and adaption by mesh refinement may substantially reduce the error from the true solution. However, modeling moving boundary-type problems requires the actual movement of points. Therefore, point movement adaption is the key to the gridless method. Other adaption methods are only included with point movement adaption as necessary to maintain solution accuracy.

#### **Goals for Implementing Adaption**

To keep the adaption method simple, and to allow for a smooth transition to multi-dimensional models, four goals were followed for implementing adaption into the gridless

method. The first goal required that the adaption be semi-autonomous. To achieve this goal, suggested default values were developed for most of the programs parameters. The program user was required only to enter the convergence parameter ( $\epsilon$ ), the point movement parameter ( $\alpha$ ), and the point refinement parameter ( $\gamma$ ). For global adaption, the global movement parameter ( $\alpha_{\text{global}}$ ) was required.

The second goal was to incorporate generality to permit the solution of a wide class of problem sets. The Fortran code was developed such that changing the governing equation only required modifying one subroutine. Therefore, nearly any one-dimensional problem may be solved by the gridless program with no change to the program itself. This is possible because of the modularity within the gridless technique's logic.

Third, the adaption method must be sufficiently robust to achieve a similar solution regardless of the initial number of points or point locations. This capability was verified directly by numerical experiments in Chapters 5 and 7.

The final goal was to allow generality in the technology to accommodate multi-dimensional problems. To make the transition to higher dimension models, the adaption techniques were kept very simple. By avoiding unnecessary complexity in the one-dimensional model, the methods developed are much more likely to apply in higher dimensions. In addition, several ideas and combinations were studied rather than limiting the research to the methods that provided the best one-dimensional solution.

### **Multidimensional Gridless Adaption**

In a multidimensional spring analogy, a single point's position is affected by all other points in the domain. Because of the complexity of this system, a global adaption method that simultaneously finds a point location for every point in a single iteration seems most appropriate. The greatest difficulty should be in determining the region that each point occupies.

The multi-dimension spring system should still utilize a similar adaption stiffness coefficient ( $k$ ). However, the effect of the cross derivative terms should be investigated. Similar convergence and movement parameters should also be used.

## Summary

The chapters demonstrate some of the capabilities of the gridless numerical solution technique. The information gained from this study will be a springboard for the follow-up work to be performed in multi-dimensions. Hopefully, the gridless approach will resolve the problems and inefficiencies encountered with the adaption of structured and unstructured grid methods. In time, this method could develop into a powerful tool for accurately modeling complex geometries and moving boundary problems, and for problems involving the dynamic coupling of a fluid and structure.

Blank Page

## REFERENCES

- Anderson, Dale A., John C. Tannehill, and Richard H. Pletcher. 1984. Computational fluid mechanics and heat transfer. Bristol, Penn.: Taylor and Francis.
- Batina, John T. 1992. A gridless euler/navier-stokes solution algorithm for complex-two dimensional applications. Hampton, Virg.: National Aeronautics and Space Administration. NASA Technical Memorandum 107631.
- Batina, John T. 1993. A gridless euler/navier-stokes solution algorithm for complex-aircraft applications. Reno, Nev.: American Institute of Aeronautics and Astronautics. AIAA 93-0333.
- Demkowicz, L., J. T. Oden, W. Rachowicz, and O. Hardy. 1989. Toward a universal h-p adaptive finite element strategy, parts 1, 2 and 3. Mechanics and Engineering 77: 79-212.
- George, P. L. 1991. Automatic mesh generation. New York: John Wiley and Sons.
- Gui, W., and I. Babuška. 1986. The h, p, and h-p versions of the finite element in one dimension, parts 1, 2, and 3. Numerische Mathematik 49: 566-683.
- Kincaid, David, and Ward Cheney. 1991. Numerical analysis. Pacific Grove, Calif.: Brooks/Cole Publishing Company.
- Liszka, T., and J. Orkisz. 1980. The finite difference method at arbitrary irregular grids and its applications in applied mechanics. Computers and Structures 2: 83-95.
- Logan, J. David. 1987. Applied mathematics. New York: John Wiley & Sons.
- Poulikakos, D. 1994. Conduction heat transfer. Englewood Cliffs, N. J.: Prentice-Hall.
- Wolfe, Walter P., James M. Nelsen, Roy S. Baty, Glenn A. Laguna, Frank J. Mello, Christine E. Hailey, and N. Todd Snyder. 1996. A gridless technique for fluid/structural dynamic coupling on flexible membranes. Albuquerque, N.M.: Sandia National Laboratories. SAND96-0199.

## DISTRIBUTION

### MS

1 1088 A. L. Thornton, 3600  
1 0841 P. J. Hommert, 9100  
1 0828 R. D. Skocypec, 9102  
Please route to 9104, 9111,  
9114, 9115  
1 0825 R.S. Baty, 9115  
1 0836 C. W. Peterson, 9116  
Please route to 9112, 9113  
1 0836 J. H. Strickland, 9116  
1 0836 W. P. Wolfe, 9116  
1 0437 E. P. Chen, 9118  
Please route to 9117  
1 0951 G. A. Laguna, 9621  
1 9018 Central Technical Files, 8523-2  
5 0899 Technical Library, 4414  
1 0619 Print Media, 12615  
2 0100 Document Processing, 7613-2  
For DOE/OSTI

### EXTERNAL

1 James M. Nelsen  
Precision Fabrics Group  
New Mexico Office  
PO Box 90  
Cedar Crest, New Mexico 87008  
1 Dr. Glenn Gebert  
Sverdrup Technologies, Inc.  
TEAS Group  
PO Box 1935  
Eglin AFB, FL 32542  
5 Dr. Christine E. Hailey  
Department of Mechanical and  
Aerospace Engineering  
Utah State University  
Logan, Utah 84322-4130  
5 N. Todd Snyder  
Department of Mechanical and  
Aerospace Engineering  
Utah State University  
Logan, Utah 84322-4130