COLLOCATION METHODS FOR THE NUMERICAL BIFURCATION ANALYSIS OF SYSTEMS OF NONLINEAR PARTIAL DIFFERENTIAL EQUATIONS

HAMID SHARIFI

A THESIS
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE & SOFTWARE ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
CONCORDIA UNIVERSITY
MONTRÉAL, QUÉBEC, CANADA

SEPTEMBER 2005

Your file Votre référence
ISBN: 0-494-09969-0
Our file Notre référence
ISBN: 0-494-09969-0

Abstract

Collocation Methods for the Numerical Bifurcation Analysis of Systems of Nonlinear Partial Differential Equations

Hamid Sharifi, Ph.D.
Concordia University, 2005

The study of nonlinear phenomena has been an important endeavor for scientists. Some nonlinear phenomena can be modeled mathematically as nonlinear partial differential equations (PDEs). There are no analytical solutions for most nonlinear PDEs. Therefore, an appropriate numerical method must be used in order to compute an adequate approximate solution.

A new class of numerical methods, called Finite Element Collocation Methods with Discontinuous Piecewise Polynomials, has recently been proposed for solving nonlinear elliptic PDE. In this thesis, this method has been generalized for solving nonlinear elliptic PDE systems using an alternative nested dissection solution procedure. Using a modified formulation of the pseudo-arclength continuation method, we have used this method in continuation studies and in the numerical bifurcation analysis of nonlinear PDE systems. In the thesis the method is introduced gradually, starting with the simplest case, linear ODE BVPs, followed by nonlinear ODE BVPs, linear scalar PDEs, nonlinear scalar PDEs, continuation problems in nonlinear scalar PDEs, and, finally, continuation problems for systems of nonlinear PDEs.

AUTO has probably been the most widely used continuation software package for ODE problems. The collocation method introduced in this thesis, as well as the numerical method used to solve the resulting systems of nonlinear equations, can be viewed as a generalization to PDEs of the robust and powerful techniques for ODE BVPs that have made AUTO so widely used in computations and as a model for other continuation software projects.

As a part of the research toward the construction of an AUTO-like software package for PDE problems, prototype software has been developed for the numerical bifurcation analysis of nonlinear elliptic PDE systems in two-dimensional space (2D). The UML (The Unified Modelling Language) notation is used to present the implementation algorithms and our object-oriented prototype software.

We consider several test problems, as well as some practical applications, such as the Bratu-Gelfand problem, the Brusselator system, and the streamfunction-vorticity formulation of the Navier-Stokes equations for a two-dimensional incompressible fluid flow problem. These examples demonstrate the capabilities and the strength of the collocation method with discontinuous elements for solving substantial PDEs continuation problems.

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

In His Name, The Most High

Acknowledgments

I would like to express my utmost gratitude and appreciation to my dear supervisor, Professor E.J. Doedel, whose profound knowledge, understanding, patience, kindness, financial support and especially his wise directions, greatly helped me to achieve the best results throughout the years of my research. Without his guidance and advice the present work would have not been fulfilled.

I am further grateful to Professor M. J. Friedman who accepted to become the external examiner, and I am also grateful to Professor G. H. Vatistas who accepted to be the Concordia external examiner of this thesis. I would like also to genuinely thank professor T. D. Bui and professor T. G. Fevens who verified the content of my thesis, as well.

Finally, I thank my wife, Dr. K. Dalir, and my parents for their support and love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

For more than a century, solving systems of nonlinear partial differential equations has received special attention in science and in engineering, and research on nonlinear phenomena has been an important endeavor for scientists. We encounter nonlinear behaviour in Physics, Chemistry, Biology, Engineering, Social Science, and so on. Some nonlinear phenomena can be modeled mathematically as nonlinear partial differential equations (PDEs). For example, fluid flow can be modeled mathematically as a system of nonlinear PDEs, known as the Navier-Stokes equations. Turbulent vortex motion, Taylor-vortex flow and vibrations of columnar vortices are examples of fluid flow problems, which can be represented by partial differential equations that are special cases of the Navier-Stokes equations.

Analytical solutions can generally be found only for simple forms of PDEs, defined over simple geometrical domains. There are no analytical solutions for most nonlinear PDEs. As a result, an appropriate computational or numerical method must be found in order to compute an adequate approximate solution. Progressive

development of more and more powerful computers has made numerical methods omnipresent. There exist several classes of numerical methods for solving PDEs. In particular, finite difference methods, finite element methods, and boundary element methods are used extensively for solving linear and nonlinear PDEs.

Collocation methods, although not as widely used as methods mentioned above, are another class of numerical methods. A class of collocation methods, called *Piecewise Polynomial Collocation*, is widely used for solving Boundary Value Problems (BVPs) in Ordinary Differential Equations (ODEs). This type of collocation method has the advantages of high accuracy, known mesh adaptation strategies, and efficient solution procedures for the associated linear systems. For difficult ODE problems, the collocation method is considered as the method of choice, for example, see [44] for some representative applications. *COLSYS* and its subsequent version *COLNEW* are well-known computer programs for scientific and engineering computations based on the collocation method. They solve multi-point boundary value problems for mixed order systems of ordinary differential equations. We will show in this thesis that the piecewise polynomial collocation method is also an appropriate method for solving PDEs.

An important aspect of ODE or PDE analysis is the prediction of solutions as system parameters change. Variation of these parameters, sometimes called control parameters, results in the variation of the behaviour of system. For example a control parameter could be the Reynolds number, or the temperature or the density, in fluid flow problems. If a control parameter changes, then we often find a unique continuous solution family. However, sometimes the solution changes suddenly and dramatically when a control parameter passes through a critical value. For example, a rod under compression load will buckle when the load passes a critical value. These types of problems are often called *bifurcation problems.* In a bifurcation analysis, we study the behaviour of solutions, their stability, and changes of other properties, as one or more control parameters of the system are varied.

Continuation methods are commonly used in a numerical bifurcation analysis. *Continuation of solutions*, also called *path following*, is an important algorithm in the numerical analysis of nonlinear equations. Some important critical points along solution families are *folds* (or *limit points*), *bifurcation points* (or *branch points*) and

Hopf bifurcation points. They lead to qualitative change in the solution behaviour, and it is therefore important to locate such points in a bifurcation analysis.

There are several continuation software packages available, for example, *AUTO*, *CONTENT*, *DDE-BIFTOOL*, *PDECONT*, *MULTIFARIO*, *LOCA* and *MATCONT* (see Section 2.3.2). AUTO has probably been the most widely used continuation software package for ODE problems. Several other ODE continuation software packages are based on AUTO, although AUTO remains, by far, the most efficient. The AUTO package, originally developed by E. J. Doedel, is based on the piecewise polynomial collocation method, and is capable of doing a bifurcation analysis for systems of ordinary differential equations (ODEs), subject to boundary and integral constraints.

As PDEs appear in many engineering and scientific problems, there is a need to construct an AUTO-like software capable of numerical bifurcation analysis of PDE systems. Developing such continuation software package for nonlinear PDEs problems has been the objective of several researchers in the past thirty years. E. J. Doedel has presented some interesting results for the solution of elliptic partial differential equations by the method of *collocation with discontinuous piecewise polynomials.* The present thesis is the continuation of this work for the bifurcation analysis of nonlinear elliptic PDEs system.

When the complexity of a software package is considerable, a powerful modeling technique is crucial. An acceptable modeling language must contain: *model elements* (fundamental modeling concepts and semantics), *notation* (visual presentation of model elements) and *guidelines* (idioms of usage within the context). As a bifurcation software is complex software, we have used an object-oriented modeling language in this thesis to present our implementation algorithms and our prototype software. We have selected *the UML (The Unified Modeling Language)* , as our object-oriented modeling language. UML is considered as a standard notation for specifying, visualizing, constructing, and documenting an object-oriented model.

1.2 Objectives of the thesis

In the literature, we can find much work on the finite element method and on object oriented software packages for PDEs. For example, we mention papers written

by Dubois-Pèlerin, Zimmermann and Bomme [59, 153], Forde et al. [70], Kong and Chen [90], Menétry and Zimmermann [101], Quézel, Fafard and Fortin [116], Scholz [131] and Zeglinkski, Han and Aitchison [154], Sharifi and Gakwaya [135], as well as the master's thesis of Sharifi [134]. Corresponding literature and object oriented software for collocation methods is much more limited.

The work by Doedel [45] on elliptic PDEs shows that collocation with discontinuous piecewise polynomials is an viable technique for solving nonlinear PDEs. Further development of this work can be found in an article by Doedel and Sharifi [55], where the method is used for continuation problems in nonlinear elliptic PDEs. This thesis is the continuation of this work, extending it to the numerical bifurcation analysis of *systems* of nonlinear elliptic PDEs. The main objective of the thesis is to present the collocation method as an excellent technique for solving nonlinear PDE systems. Another important objective of this thesis is to develop a prototype AUTO-like continuation package for systems of PDEs.

In the thesis, we will restrict attention to the numerical analysis of nonlinear elliptic PDEs in two-dimensional space ($2D$). Our method can also be applied to $3D$ problems, but as the final discrete system is large, iterative methods need to be used. Although an important topic, iterative techniques (apart from the Newton method) will not be considered in this thesis. For investigating the applicability and efficiency of the method, we consider various applications, including the Navier-Stokes equations. In the case of the Navier-Stokes equations, the classical finite difference method is difficult to apply and the solution of the system that results from this type of discretisation requires much execution time[1].

Specifically, the main objectives of the thesis are:

- Generalization of the method of "collocation with discontinuous piecewise polynomials" to cover the solution of *systems of nonlinear elliptic PDEs*. We develop an algorithm that can be used to solve PDE systems such as the Navier-Stokes

[1]In the case of simple (low order) finite differences, a fine mesh is needed that eventually gives a large system of equations. In case of higher order finite differences we encounter the problem of how to apply boundary conditions.

equations. As boundary conditions, Dirichlet, Neumann, periodic or a combination of these types are considered.

- Development of numerical continuation and bifurcation methods in conjunction with the collocation method for the *bifurcation analysis* of systems of PDEs. Use of Keller's pseudo-arclength equation [87, 47, 49, 50] is fundamental in this method. Part of this work was already presented in a paper by Doedel and Sharifi [55].

- Construction of a *modified formulation of the pseudo-arclength equation.* In this formulation the step size and the direction vector can be adjusted in each Newton's iteration. The original algorithm has problems to detect bifurcation points when boundary conditions depend on a control parameter.

- An *alternative formulation of the nested dissection method* for constructing the global system from the elementary equations of each element. The original algorithm presented in [45] cannot be used in general, as there are cases in which it fails. Specifically, there are cases where one of key matrices (E in Equation (3.37)) can be singular, for example, at folds and branch points.

- Presentation of an *object-oriented modelling* of the above method, including the definition of classes, correctness conditions, *etc.*, using UML notation. This makes the method more understandable, and it also makes further development of the software easier.

- Use of object-oriented programming with C++ to develop a *software package* for the bifurcation analysis of systems of nonlinear elliptic PDEs.

- *Application of the algorithms* presented in the thesis and implemented in the software to problems including

  - the Bratu-Gelfand problem,
  - the Brusselator problem, used in biochemical reaction models,
  - the steady state Navier-Stokes equations, which arise in fluid flow problems.

Convergence and numerical stability is examined for linear ODE problems, but will not be emphasized for PDEs in this thesis. We want to demonstrate the practical feasibility of the new method for PDE systems, leaving its complete theoretical treatment for future studies. Nevertheless, for PDEs, we present some stability results, and we demonstrate the accuracy of the method for specific applications.

1.3 Outline of the thesis

Background material is given in Chapter 2, including a brief introduction to the finite difference method, the finite element method, and the collocation method, as well as a short presentation of continuation and bifurcation methods, and software. Use of UML notation to present an object-oriented model is also discussed in this chapter.

In Chapter 3, in preparation for the more complicated case of nonlinear PDEs, we consider the finite element collocation method for linear ODEs. We discuss various aspects of the piecewise polynomial collocation method. We explain the original nested dissection solution procedure, and a simple mesh generation procedure which is suitable for this method. The finite difference equivalence of the collocation method, basis functions, complexity, as well as the stability and the convergence theory, are covered in this chapter.

In Chapter 4, we consider the collocation method for nonlinear BVP ODEs. We present essentially the same topics as in Chapter 3, but we present them for the nonlinear case. We explain how we can construct our global linearized system and how we can solve it using Newton's method. A modified version of the nested dissection method is also introduced here.

In Chapter 5, we present the collocation method for solving a linear elliptic PDE boundary value problem. We present the collocation method with discontinuous piecewise polynomials and its finite difference formulation, passing through similar steps as in the linear ODEs case. We also consider the stability of our collocation method, using tools from linear algebra.

The collocation method for solving a nonlinear elliptic PDE boundary value problem is presented in Chapter 6. Having already considered the linear and the

nonlinear ODE cases as well as the linear PDE case, we consider the collocation method for the nonlinear PDE case. In particular, we extend the solution procedure of Chapter 4, to cover the PDE case.

General concepts related to the numerical continuation of solutions are presented in Chapter 7. We consider parametric and pseudo-arclength continuation. The definition of a fold and a branch point, as well as procedures which are used to detect them, are discussed. We consider the computation of bifurcation direction vectors, and the branch switching procedure. The modifications which we have made in the original pseudo-arclength method are also discussed in this chapter.

In Chapter 8, we show how our collocation method can be used for continuation problems in nonlinear elliptic PDEs. Having presented collocation with discontinuous piecewise polynomials for solving nonlinear elliptic PDEs in Chapter 6, we now use this method for continuation problem.

Chapter 9 is the generalization of the procedure of Chapter 8 to the case of *systems* of nonlinear PDEs. In this chapter, we describe the modifications that are necessary in the algorithm of the preceding chapter.

In Chapter 10, we consider Object-oriented modelling and implementation of our prototype software. We use different UML models and diagrams, such as use case, packages, classes, interaction, states and activities diagrams, to present our object-oriented model for the bifurcation analysis of systems of nonlinear elliptic PDEs. We present our prototype software in the C++ language.

Chapter 11 is dedicated to numerical applications of our prototype software. After presenting test applications, *i.e.*, PDEs with known exact solutions, we consider applications such as the Bratu-Gelfand problem, the Brusselator system, and the streamfunction-vorticity formulation of the Navier-Stokes equations for a two-dimensional incompressible fluid flow problem. These examples demonstrate the capabilities and the strength of the collocation method with discontinuous elements for solving PDEs continuation problems.

In Chapter 12, we give some final conclusions.

1.4 On the organization of this thesis

The finite element collocation method introduced in this thesis applies, in principle, to systems of elliptic PDEs in very general domains in any space dimension. It applies, also in principle, to finite elements of any shape. The solution algorithm, namely the nested dissection method, is suggested by the equivalence of the collocation method to a generalized finite difference method for the case of linear problems. The nested dissection procedure can also be used, in principle, for very general elliptic equations. In fact, it would have been possible to present the method immediately in its most general form. Instead, however, the decision was taken to introduce the method gradually, starting with the simplest case, linear ODE BVPs, followed by nonlinear ODE BVPs, linear scalar PDEs, nonlinear scalar PDEs, continuation problems in nonlinear scalar PDEs, and, finally, continuation problems for *systems* of nonlinear PDEs.

The decision to introduce the methods gradually, has resulted in a longer document than would have been the case otherwise. The reasons behind the particular organization of this thesis are, however, multiple, as explained below.

First, it is interesting to see the remarkable unity of the method, by reconsidering the ODE BVP case in a new light.

Second, it is important to treat the linear case separately, emphasizing its equivalence to a generalized finite difference method, as the latter suggests the use of nested dissection in a very natural way. Thereafter, the algorithmic approach for the nonlinear case will be more easily understood; in particular the nested dissection algorithm will appear less "magical" in this case.

Third, the application to continuation and bifurcations problems in elliptic PDEs introduces various subtle difficulties, that may not be appreciated by a reader who is mainly interested in the basic ideas of the algorithms for the case of PDEs. On the other hand, it is important to specify how these difficulties can be effectively addressed, in particular for the benefit of researchers and developers who wish to develop "production" bifurcation software, *e.g.*, using adaptive triangulations, based on our collocation method.

Similarly, the application of the method of collocation with discontinuous elements to the continuation and bifurcation analysis of *systems* of nonlinear PDEs introduces a new level of complexity at the implementation level, that we have chosen to describe separately.

In view of the organization of this thesis, as explained above, the following reading paths can be suggested:

- The main ideas only: Chapters 3, 5.
- A new view of collocation for ODE BVPs, and its extension to simple PDEs: Chapters 2, 3, 4, 5.
- For experts in numerical methods for PDEs, with an introduction to continuation and bifurcation methods: Chapters 5, 6, 7, 8.
- For experts, who wish to create production software for the continuation and bifurcation analysis of general systems of nonlinear elliptic PDEs: Chapters 5, 6, 7, 9, 10.

Chapter 2

Background Material

2.1 Discretization methods

Most numerical methods have been invented in response to the need for solving physical problems in engineering and science. Specifically, methods have been developed for problems for which we do not have analytical solutions. For example, the flow of a fluid can be modeled mathematically by a system of nonlinear partial differential equations (PDEs), known as Navier-Stokes equations [16, 67], for which an analytical solution has not been found in the general case. Turbulent vortex motion [7], Taylor-vortex flow [105, 146], and vibrations of columnar vortices [89, 150, 152] are examples of interesting fluid flow problems, for which scientists would like to have an appropriate numerical solutions.

There exist several classes of numerical methods that can be used for solving differential equations. Finite difference methods, finite element methods, boundary element methods, and collocation methods, are some of the numerical methods used in engineering and science. Below, we briefly discuss some of these numerical methods for solving ODEs and PDEs.

2.1.1 The finite difference method

The finite difference method is a classical method for solving differential equations. There are numerous publications that consider finite difference approximations; for example, the books of Collatz [32], Keller [83, 85], Gear [72], and Ascher, Mattheij and Russell [12]. The general theory of difference methods for boundary value problems in ODE is contained in articles by Grigorieff [76] and Kreiss [91]. Construction of generalized finite difference methods has been considered by Birkhoff and Gulati [19], Swartz [145], Keller and Pereya [88], Osborne [111, 112, 113], and Doedel [17, 40, 41, 42, 53, 54, 63, 64].

Basic finite difference approximations can be derived easily for ODEs and PDEs. As an example, consider a PDE operator that acts on a function $u(x)$ defined over a domain Ω, with given boundary conditions. To determine an approximation to the solution $u(x)$, the domain of problem must be replaced by a suitable *grid* (or *mesh*). We write the PDE at each mesh point of the grid, and each term of the PDE containing u or its derivatives is replaced by a finite difference approximation term that contains the values of u at that point and at neighboring grid points. In other words, a finite difference equation is obtained by substitution of approximated terms into the PDE. In the case of time-dependent PDEs, the finite difference approximation can be used for the time variable or for both, space and time variables [108].

Let $u(x)$ represent a function of one variable. Assume $u(x)$ is sufficiently smooth, *i.e.*, we can differentiate it several times, and each derivative is a well-defined, continuous function in an interval containing a particular point of interest x_0. u' can be approximated by the following equation, using the values of u at x_0 and x_1 (a point near x_0 at distance h).

$$u'(x_0) \approx u'(x_0)_+ = \frac{u(x_0+h) - u(x_0)}{h}. \tag{2.1}$$

The Equation (2.1) is a one-sided approximation to u'. Another one-sided approximation to u' is

$$u'(x_0) \approx u'(x_0)_- = \frac{u(x_0) - u(x_0-h)}{h}. \tag{2.2}$$

Equations (2.1) and (2.2) are first order accurate approximations to u', written $\mathcal{O}(h)$, *i.e.* the value of the error is proportional to h. A second order accurate formula is

the centered approximation:

$$u^{'}(x_0) \approx u^{'}(x_0)_c = \frac{u(x_0+h) - u(x_0-h)}{2h} = \frac{1}{2}\left(u^{'}(x_0)_- + u^{'}(x_0)_+\right). \tag{2.3}$$

Here, the error $u^{'}(x_0)_c - u^{'}(x_0)$ is $\mathcal{O}(h^2)$, as can be shown easily by Taylor expansion of $u(x_0+h)$ and $u(x_0-h)$. A third order accurate approximation ($\mathcal{O}(h^3)$) is given by [96]:

$$u^{'}(x_0) \approx \frac{2u(x_0+h) + 3u(x_0) - 6u(x_0-h) + u(x_0-2h)}{6h}. \tag{2.4}$$

A general method to derive a finite difference approximation is based on polynomial interpolation. We interpolate the function $u(x)$ by a polynomial $p(x)$ and then we use $p^{'}(x_0)$ to approximate $u^{'}(x_0)$. To find the polynomial coefficients, we interpolate $u(x)$ at some *mesh points*. For example, interpolate $u(x)$ using a third order polynomial $p(x) = a_0 + a_1 x + a_2 x^2$, with interpolation points x_0, x_0+h and x_0+2h. We have

$$\begin{aligned}
u(x_0) &= a_0 + a_1 x_0 + a_2 x_0^2, \\
u(x_0+h) &= a_0 + a_1(x_0+h) + a_2(x_0+h)^2, \\
u(x_0+2h) &= a_0 + a_1(x_0+2h) + a_2(x_0+2h)^2,
\end{aligned}$$

or

$$\begin{pmatrix} u(x_0) \\ u(x_0+h) \\ u(x_0+2h) \end{pmatrix} = \begin{pmatrix} 1 & x_0 & x_0^2 \\ 1 & x_0+h & (x_0+h)^2 \\ 1 & x_0+2h & (x_0+2h)^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix}.$$

We can solve for a_0, a_1, a_2 in terms of $u(x_0), u(x_0+h), u(x_0+2h)$, which gives

$$a_0 = \frac{1}{2}\frac{(x_0^2 + 3x_0h + 2h^2)u_0}{h^2} - \frac{x_0(x_0+2h)u_1}{h^2} + \frac{x_0(x_0+h)u_2}{2h^2}, \tag{2.5}$$

$$a_1 = -\frac{1}{2}\frac{(2x_0+3h)u_0}{h^2} + \frac{2(x_0+h)u_1}{h^2} - \frac{1}{2}\frac{(2x_0+h)u_2}{h^2}, \tag{2.6}$$

$$a_2 = \frac{1}{2}\frac{u_0}{h^2} - \frac{u_1}{h^2} + \frac{u_2}{2h^2}. \tag{2.7}$$

Evaluating $p^{'}(x_0)$, we have

$$p^{'}(x_0) = a_1 + 2a_2x_0,$$

or

$$\begin{aligned} p'(x_0) &= -\frac{1}{2}\frac{(2x_0+3h)u_0}{h^2} + \frac{2(x_0+h)u_1}{h^2} - \frac{1}{2}\frac{(2x_0+h)u_2}{h^2} \\ &+ 2(\frac{1}{2}\frac{u_0}{h^2} - \frac{u_1}{h^2} + \frac{u_2}{2h^2})x_0, \end{aligned}$$

from which

$$p'(x_0) = -\frac{3}{2h}u_0 + \frac{2}{h}u_1 - \frac{1}{2h}u_2.$$

This elementary example contains the idea of a relation between the finite difference and the collocation method (*cf.* Section 2.1.3). We will consider this relation in detail in Section 3.7, for ODEs, and in Section 5.4, for PDEs.

Second order derivatives can be approximated in the same manner. For example, the well-known central approximation, which is $\mathcal{O}(h^2)$, is given by

$$u''(x_0) \approx \frac{u(x_0-h) - 2ux_0 + u(x_0+h)}{h^2}. \tag{2.8}$$

2.1.2 The finite element method

The Finite Element Method (FEM) is perhaps the most widely used numerical method for solving PDEs. There are a large number of publications related to this method and its various applications in different engineering problems. For example, we mention the books of Zienkiewicz [155], Batoz and Dhatt [15], Dhatt and Touzot [36], Segerlind [132] and Ciarlet [31]. The use of this method in scientific computation is so extensive that several research groups work on the construction of the object oriented modeling and the software development of this method for use in different fields. See, for example, the papers by Dubois-Pèlerin, Zimmermann and Bomme [59, 153], Forde et al. [70], Kong and Chen [90], Menétry and Zimmermann [101], Quézel, Fafard and Fortin [116], Scholz [131] and Zeglinkski, Han and Aitchison [154].

Consider a PDE that involves a function $u(x)$ defined for all x in a domain subject to certain boundary conditions. The objective of the FEM is to obtain an approximation to the function $u(x)$. Like the finite difference method, the FEM needs a discretisation of the domain, namely, the domain is subdivided into sub-regions or cells, called *elements*. As an example, a two-dimensional domain can be divided into a

set of triangles or quadrilateral elements. The function $u(x)$ is approximated on each element. For example, $u(x)$ can be approximated by a linear function on each triangle or rectangle. The FEM formulation can be summarized as follows: "Projection of the *variational form* of the differential equation into a finite-dimensional function space" ([155] or [100]). Consider the following elliptic PDE

$$Lu = -\nabla \cdot (a\nabla u) + bu + c = 0, \qquad a, b, c, u \in \Omega, \tag{2.9}$$

where Ω is a bounded domain and a, b, c are sufficiently smooth functions. We can consider, for example, the following boundary conditions :

- *Dirichlet*: $u = e$ on the boundary $\delta\Omega$,
- *Generalized Neumann*: $\overrightarrow{n} \cdot (a\nabla u) + du = e$ on $\delta\Omega$,
- Mixed: a combination of Dirichlet, over $\delta\Omega_1$, and generalized Neumann, over $\delta\Omega_2$. $\delta\Omega_1 \bigcup \delta\Omega_2 = \delta\Omega$,

where $\overrightarrow{n}$ is the outward unit normal, and d, e are smooth functions on the boundary $\delta\Omega$.

Assume that u is the solution of the differential equation. We can construct the weak form by multiplying Equation (2.9) by an arbitrary *test function* v, and integrating over Ω.

$$\int_\Omega -\nabla \cdot (a\nabla u)v + buv + cv \; dx = 0. \tag{2.10}$$

We integrate by parts, *i.e.* we use Green's formula, to have

$$\int_\Omega (a\nabla u) \cdot \nabla v + buv \; dx - \int_{\delta\Omega} \overrightarrow{n} \cdot (a\nabla u)v \; ds + \int_\Omega cv \; dx = 0. \tag{2.11}$$

The boundary integral can be replaced using the boundary condition. For example, using the Generalized Neumann condition above, we have

$$\int_\Omega (a\nabla u) \cdot \nabla v + buv \; dx. - \int_{\delta\Omega} (-du + e)v \; ds + \int_\Omega cv \; dx = 0. \tag{2.12}$$

The original problem can now be replaced by the following problem: Find u such that

$$\int_\Omega (a\nabla u) \cdot \nabla v + buv + cv \; dx - \int_{\delta\Omega} (-du + e)v \; ds = 0 \qquad \forall v. \tag{2.13}$$

This equation is called the *variational* or *weak* form of the differential equation. Any solution of the differential equation is also a solution of the variational problem. The solution of the variational problem is also called the *weak solution* of the differential equation.

Assume that the solution u and the test functions v belong to a function space V. We select an N-dimensional subspace $V_N \subset V$. Projecting the variational form of the differential equation onto a finite-dimensional function space means finding u and v in V_N instead of V. If the space V_N tends to V as N goes to ∞, the approximate solution is expected to converge toward the exact solution. As the differential operator is linear here, we satisfy the variational equation for N linearly independent test-functions $\Phi_i \in V_N$ (the basis of V_N), *i.e.*,

$$\int_\Omega (a\nabla u)\cdot\nabla\Phi_i + bu\Phi_i + c\Phi_i \; dx - \int_{\delta\Omega}(-du+e)\Phi_i \; ds = 0 \qquad i = 1,\cdots,N. \tag{2.14}$$

Expressing the solution u in terms of the test-functions of V_N

$$u(x) = \Sigma_{j=1}^{N} U_j \Phi_j(x), \tag{2.15}$$

we find the following system of equations

$$\begin{aligned} &\Sigma_{j=1}^{N}\left(\textstyle\int_\Omega (a\nabla\Phi_j)\cdot\nabla\Phi_i + b\Phi_j\Phi_i \; dx + \int_{\delta\Omega} d\Phi_j\Phi_i \; ds\right) U_j \\ &\quad = -\textstyle\int_\Omega c\Phi_i \; dx - \int_{\delta\Omega} e\Phi_i \; ds, \qquad\qquad i = 1,\cdots,N. \end{aligned} \tag{2.16}$$

Taking:

$$\begin{aligned} K_{i,j} &= \textstyle\int_\Omega (a\nabla\Phi_j)\cdot\nabla\Phi_i \; dx, \qquad \text{(Stiffness matrix)} \\ M_{i,j} &= \textstyle\int_\Omega b\Phi_j\Phi_i \; dx, \qquad\qquad \text{(Mass matrix)} \\ P_{i,j} &= \textstyle\int_{\delta\Omega} d\Phi_j\Phi_i \; ds, \\ F_i &= -\textstyle\int_\Omega c\Phi_i \; dx, \\ H_i &= \textstyle\int_{\delta\Omega} e\Phi_i \; ds, \end{aligned} \tag{2.17}$$

we can write the system 2.16 in the form

$$(K+M+P)U = F+H. \tag{2.18}$$

or more compactly,

$$AU = B.$$

Here $A = K+M+P$ and $B = F+H$, where K, M and P are N by N matrices, and F and H are N-dimensional vectors.

For a self-adjoint problem, the matrix A is symmetric and positive definite. This is characteristic for many elliptic problems in engineering and applied science; specifically for those problems that can be considered as minimization problems.

In the finite element procedure, after the discretization of the domain, Equation (2.16) is written for each element, called *the elementary equations*. These equations are *assembled* in a global system, which then must be solved.

The interpolation of the solution inside each element is done in terms of *nodal* values. There are several choices of the test-function space, and selecting a suitable space is an important task. The space of continuous polynomials is the most common test-function space (V_N). A usual basis for V_N is the set of functions Φ_i which are linear on each element and take the value 0 at all nodes x_j, except one, say x_i. The property $\Phi_i(x_i) = 1$ guarantees that

$$u(x_i) = \Sigma_{j=1}^{N} U_j \Phi_j(x_i) = U_i.$$

As the elementary equations are in terms of nodal values, upon solving the global FEM system we obtain the nodal values of the approximate solution. The basis function Φ_i of an element, say l, is zero on all other elements and it does not need to be considered in the calculation of elementary equations of other elements. As a result, in the integration of $K_{i,j}$, $M_{i,j}$, $P_{i,j}$, F_i and H_i of an element l, we need only compute the integral over the element l. Thus $K_{i,j}$ and $M_{i,j}$ are zero except when x_i and x_j are belong to the same element. Therefore K and M are banded matrices. The bandwidth of the matrix structure depends on the numbering of mesh points.

In the assembling procedure, the contributions of each element (elementary equations) are added to the global FEM system. After finding the elementary matrices, one adds their components to the corresponding positions in the global matrices or vectors, using the *connectivity* data of the mesh. The connectivity data is usually stored in a matrix structure that for each element gives its global nodal point indices. As an example, for a triangular element i takes the values $1, 2, 3$, and a matrix $Elem[l, i]$ can be used for storing the information. Here l is the index of the element, and i is the local element node number. Then the elementary matrix k of an element l can be added to the global matrix K using the following equation.

$$K_{Elem[l,m],Elem[l,n]} = K_{Elem[l,m],Elem[l,n]} + k_{m,n}, \qquad m, n = 1, 2, 3.$$

Finally, one can find the nodal values (the vector U) by solving the overall system $AU = B$, also using the boundary conditions.

2.1.3 The collocation method

Consider a general linear homogeneous elliptic differential equation

$$L(u) = 0, \qquad \text{in domain } \Omega, \tag{2.19}$$

with boundary condition

$$S(u) = 0 \qquad \text{on } \delta\Omega, \tag{2.20}$$

where $\delta\Omega$ is the boundary of the domain Ω. In a *global collocation method*, we assume that the solution u is approximately equal to a linear combination of certain linearly independent basis functions ϕ_i, $(i = 1, 2, \cdots, N)$.

$$u \simeq \alpha_1\phi_1 + \alpha_2\phi_2 + \alpha_3\phi_3 + \cdots + \alpha_N\phi_N, \tag{2.21}$$

where α_i, $i = 1, 2, \cdots, N$ are unknown coefficients, which are determined by requiring the differential equation to be satisfied at M collocation points over the domain,

$$L(u_j) = L(\sum_{i=1}^{N} \alpha_i\phi_i(x_j)) = 0, \qquad j = 1, 2, \cdots, M,$$

and, in addition, the boundary conditions must be satisfied at a discrete set of points on the boundary.

In the *finite element collocation method*, we discretise the domain Ω into elements. Over each element, we assume that the variable u is approximately equal to a linear combination of certain linearly independent basis functions ψ_{ij}, $(j = 1, 2, \cdots, n)$, that locally satisfy certain continuity requirements between adjacent elements. For an element i, we have

$$u_i \simeq \beta_{i1}\psi_{i1} + \beta_{i2}\psi_{i2} + \beta_{i3}\psi_{i3} + \cdots + \beta_{in}\psi_{in}, \tag{2.22}$$

where β_{ij}, $j = 1, 2, \cdots, n$ are unknown coefficients of the element i. In the finite element collocation method (or piecewise polynomial collocation method), the unknown coefficients β_{ij} $(i = 1, 2, \cdots, N_E,\ j = 1, 2, \cdots, n)$ are determined by requiring

the differential equation to be satisfied at m local collocation points for each element. Here N_E is the number of elements.

$$L(u_{ik}) = L(\sum_{j=1}^{n} \beta_{ij}\psi_{ij}(x_{ik})) = 0, \qquad i = 1, 2, \cdots, N_E, \ k = 1, 2, \cdots, m.$$

To obtain a square system (*i.e.*, with the same number of equations and unknowns) , we must add domain boundary conditions and continuity requirements between adjacent elements. For example, consider the one-dimensional domain $[a, b]$, subdivided into N_E intervals (elements),

$$a = x_0 < x_1 < \cdots < x_{N_E} = b.$$

For each interval $[x_{i-1}, x_i]$ (element i), we can have m collocation points

$$x_{ik} = \frac{1}{2}((x_{i-1} + x_i) + (x_i - x_{i-1})\xi_k), \qquad k = 1, 2, \cdots, m, \tag{2.23}$$

where, the ξ_k are distinct value in $(-1, 1)$. The number of collocation points in each element, which is usually the same for all elements, is chosen so that it matches the number of unknowns, also taking into account the continuity conditions at the boundaries of the element.

The finite element collocation method is at the heart of this thesis, and its different forms and applications will be explored in much detail in the following chapters.

2.2 Historical notes on collocation methods

2.2.1 Collocation for ODEs

The piecewise polynomial collocation method for ODEs was originally proposed by Russell and Shampine [126]. They developed and analyzed collocation methods for solving systems of first order boundary value problems for ordinary differential equations (ODE BVP) [124, 127]. As already mentioned, the piecewise polynomial collocation method uses a polynomial approximation for each element, where the polynomial satisfies the ODE exactly at a specific number of collocation points. This method is widely used for solving boundary value problems in ODEs. For example, it is the basic discretisation method in the software packages COLSYS [10, 9], COLDAE [13], and AUTO [48, 56]. Its advantages are high accuracy [35], known

mesh adaptation strategies [125] , and there are efficient solution procedures for solving the associated linear systems [12]. If Gauss points are chosen as collocation points then the approximate piecewise polynomial solution has super-convergence characteristics, as analyzed for BVPs in ODEs by de Boor and Swartz [35]. More precisely, in this method, the m parameters ξ_k, $k = 1, \cdots, m$ in Equation (2.23) for an element i (interval $[x_{i-1}, x_i]$) are taken to be the roots of the orthogonal polynomial of degree m in the interval $(-1, 1)$. Other locations of the collocation points have also been proposed by some authors; for example, Lanczos [94] proposed the use of Chebyshev points for global collocation methods. These points can also be used for piecewise polynomial collocation. The error at a point x of an element i having m collocation point approximation is proportional to

$$e(x) \simeq \prod_{k=1}^{m} (x - x_{ik}),$$

where $\|e\|_\infty$ is minimized when we choose Chebyshev points.

Osborne [113] presented the relationship between finite difference methods and certain collocation methods. The equivalence of collocation to a discrete Galerkin method was considered by Russell and Varah [129]. One advantage of collocation methods over finite element Galerkin methods is the calculation of coefficient matrices, as there is no integral to be evaluated. The work of Russell and Shampine on piecewise polynomial collocation for ODE BVPs was continued by several authors; for example, Ascher, Christiansen and Russell [10] use it for solving mixed order ODE BVPs.

Several authors have studied piecewise polynomial collocation methods for the numerical solutions of singular two-point boundary value problems; for example, we mention the papers by Reddien [117], Reddien and Schumaker [118], Brabston and Keller [21], de Hoog and Weiss [79], Doedel and Reddien [53].

Ascher and Bader [8] considered the enhanced stability of piecewise polynomial collocation methods. Stability considerations sometimes suggest the employment of non-symmetric collocation points instead of Gauss points. Ringhofer [121] studied the stability of a class of quasilinear singular perturbed boundary value problems, using spline collocation with specified nonsymmetric collocation points. A noncompact scheme for the reduced equation of such singular perturbation problems for boundary

value problems was proposed by Mahmood and Osborne [98]. They show that this scheme is stable when certain nonsymmetric collocation points are used.

Huang and Sloan [81] considered spectral collocation approximations and some upwinding features for pseudospectral collocation methods. They show that the main feature of an upwinding scheme is that the eigenvalues of the first-order differentiation matrix are located in the left-half plane. Houstis, Christara and Rice [80] presented a collocation method based on quadratic piecewise polynomials (splines) for second order two point boundary value problems. They obtained $O(h^{3-j})$ global error estimate for the jth derivative of the error. The standard collocation at midpoints gives $O(h^{2-j})$ error bounds. Christara and Smith [30] developed and analyzed multigrid methods for quadratic spline collocation equations arising from the discretisation of one-dimensional second-order differential equations (ODEs).

2.2.2 Collocation for PDEs

Collocation methods for parabolic equations in a single space variable were studied by Douglas and Dupont [57, 58]. Cavendish [22] and Ito [82] used a bicubic spline collocation method for solving elliptic differential equations to get an $\mathcal{O}(h^2)$ method, where h is the mesh size. They used the knots of the bicubic splines as collocation points. Prenter and Russell [115] used collocation methods for solving elliptic differential equations on a unit square. They use Gauss points as collocation points and bicubic Hermite functions as basis functions. They argue that under certain smoothness conditions this method is of $\mathcal{O}(h^4)$.

An improved C^1 finite element collocation method for elliptic equations was considered by Percell and Wheeler [114]. They used collocation at Gauss points as a means of determining a C^1 finite element approximation. They removed some hypotheses used by Prenter and Russell, such as the assumption of existence and uniqueness of the piecewise polynomial solution and the assumption that the collocation approximation satisfies certain bounds. Allen and Pinder [2] proposed collocation methods with upwinding features for a convection-dispersion transport equations. They used Hermite cubic piecewise polynomial (spline) collocation with some nonsymmetric collocation points. They have shown that this method produces oscillation-free solutions for linear problems.

Other smooth piecewise polynomial collocation methods, called *spline collocation methods* are considered by several researchers, for example, Bialecki [18], Christara [25] and Sun [140, 139]. A solution procedure for second order elliptic boundary value problems using a quadratic spline collocation discretisation scheme on rectangles was considered by Houstis, Christara and Rice [28]. They studied the partitioning and mapping of computations on a parallel computer, the NCUBE/7 (128 processors), having a hypercube architecture.

A disadvantage of the spline collocation method is that point iterative solvers like ITPACK [1] do not converge, even for simple problems [14, 61, 144]. As a result, Gauss elimination with scaling and full or partial pivoting is often used for solving the collocation system. Nevertheless, Dyksen [60] used a tensor product generalized ADI (Alternating Direction Implicit) method for separable elliptic problems to solve the collocation system. Sun [144, 138] used a FFT algorithm to solve the tensor product collocation equations. In another paper, Sun [140] used cyclic reduction and FARC algorithms, a combination of cyclic reduction and Fourier analysis, for solving a Hermite cubic spline collocation system. An algorithm for solving a class of high-order spline collocation systems, which arise from discretising the Poisson equation on a rectangular domain with Dirichlet boundary conditions, was given by the same author [141]. He also considered an eigenvalue analysis of the first-order Hermite cubic spline collocation differentiation matrices with arbitrary collocation points [142]. He also proposed a class of spline collocation methods with upwind features for solving singular perturbation problems.

Russell and Sun [128] analyzed spline collocation differentiation matrices that arise when solving PDEs with periodic boundary conditions. The error analysis of parabolic and hyperbolic partial integro-differential equations is considered by Fairweather [68] for orthogonal spline collocation and for a modified spline collocation method.

Christara [26] worked on *Quadratic Spline Collocation* (QSC) methods for linear second order elliptic PDEs. In her work, the collocation points were chosen to be the midpoints of uniform grids. She has demonstrated that this method has optimal order of convergence. Several solvers for spline collocation equations arising from the discretisation of elliptic PDEs were studied in her work. She also implemented the

QSC methods on parallel machines [27]. Constas [34] studied Fast Fourier Transforms (FFTs) for the QSC method. Christara and Sun Ng [29, 106] extend QSC to systems of two coupled linear second-order PDEs in two dimensions. They treated the PDEs system as one entity, and no decoupling was applied.

Discontinuous piecewise polynomial collocation for PDEs was introduced by Doedel [45]. One of the advantages of this method is its generality for solving a wide range of problems, ODEs as well as PDEs. This collocation method represents a type of unified method, and is the root of the methods presented in this thesis.

2.3 Continuation and bifurcation

2.3.1 Basic concepts

Nonlinear phenomena are important in engineering and science. For example, in Physics, Chemistry, Biology, Engineering, Social Science and even our daily life, we encounter nonlinear behavior. Some of these phenomena can be modeled mathematically as a nonlinear equation of the form

$$G(u, \lambda) = 0, \tag{2.24}$$

where G is a smooth function, and where λ represents system control parameters. For example λ could be Reynolds number, or a temperature or density in a fluid flow problem. If a control parameter is changed, then usually we find a unique continuous solution curve; often called a *solution branch* or a *solution family*. However, sometimes the nature of the solution changes dramatically when a control parameter passes through a critical value. For example, a rod under compression load will buckle and the stability of the system will be lost when the load passes a critical value. Such a phenomenon is called a bifurcation. When there is a bifurcation, the linear stability theory fails and it does not give us valuable information on the behaviour of our nonlinear system. As a result, we need bifurcation analysis and a more general stability theory. In a numerical bifurcation analysis, we normally use a continuation method and we study the behaviour of the solutions, their stability and variation of other properties as one or more parameters of the system change.

Folds (or limit points), bifurcation points (or branch points) (Figure 2.1) and

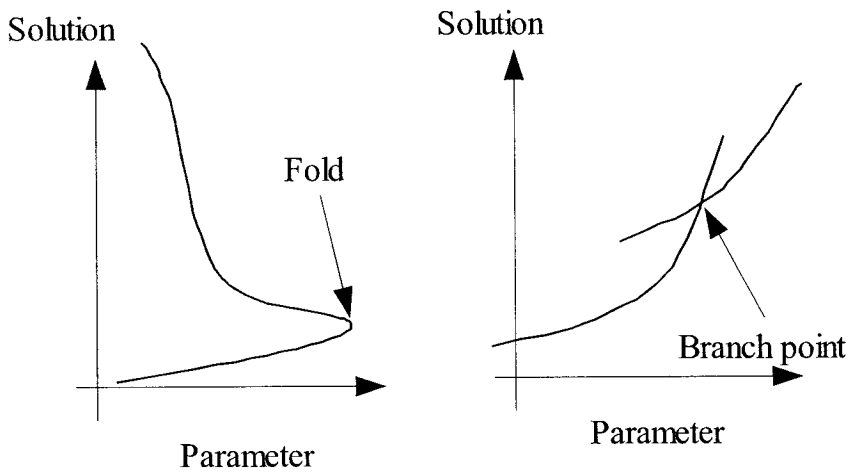


Figure 2.1: Folds and bifurcation points

Hopf bifurcation points may be found in the numerical bifurcation analysis of a system. They result in the qualitative change of behaviour of solutions. Such points can be detected during continuation of solutions.

Numerical continuation of solution families, or *path following*, is the subject of many books, for example those written by Garcia and Zangwill [71], Gould and Tolle [74], Keller [87], Rheinboldt [119], Seydel [133], Todd [148], and Allgower and Georg [4, 5, 6]. The principle of continuation methods is to compute solutions near a known solution, then adding the new solution to the known solution set, and to repeat this process. As a result, we must choose a method to compute a new solution near a known solution. In this regard, continuation methods can be divided into two categories, namely *simplicial continuation* methods and *predictor-corrector* methods.

In simplicial continuation methods, we follow a piecewise linear curve that approximates a branch of solutions. If G maps from R^n to R^{n-k}, we generally expect u to lie on a $n-k$ dimensional linear subspace, or on a $n-k$ dimensional face of a simplex in R^n [77]. In other words, in simplicial continuation we represent u_i as the intersection of a manifold with a $n-k$ dimensional simplex σ_i. Allgower and Georg used simplicial continuation methods for approximation of fixed points and solutions

to systems of equations [3]. Dobkin et al. [39] have used this method for tracing a curve that is represented as the contour of a function.

Predictor-corrector methods consist of two steps:

- The predictor step: an approximate point along the curve is computed, usually in the direction of the tangent at a known point of the solution curve.
- The corrector step: bring the predicted point to the solution curve using one or more iterations.

The most popular predictor-corrector continuation method is *pseudo-arclength continuation*, introduced by Keller [86], which is also the basic method used for continuation in this thesis.

2.3.2 Continuation and bifurcation software

There are several continuation software packages available, for example

- AUTO [56, 48, 52, 23, 51, 43] is a software package for continuation and bifurcation problems in ODE BVP,
- CONTENT [93, 92] is a multi-platform interactive environment to study dynamical systems. Part of this software is based on a C translation of a version of the AUTO package written in Fortran 77.
- DDE-BIFTOOL [62, 66, 65] is a Matlab package for the numerical bifurcation analysis of delay differential equations with several fixed discrete delays. It allows the computation, continuation and stability analysis of steady state solutions, and compute Hopf and Fold bifurcations and periodic solutions. DDE-BIFTOOL is also based on the piecewise polynomial collocation discretisation method.
- MULTIFARIO [78] is a set of subroutines and data structures for computing multidimensional solution manifolds.
- PDECONT [122, 97] is a Newton-Picard single shooting software. It implements the Newton-Picard single shooting algorithm for the continuation of periodic solutions of large-scale problems.

- LOCA [130] is a library of continuation algorithms that enables the tracking of solution branches as a function of system parameters, and the direct tracking of bifurcation points. Algorithms are chosen to work for large scale problems, and to run on distributed memory parallel machines.

- MATCONT [38, 37] is a Matlab implementation of a number of algorithms from AUTO and CONTENT, with additional capabilities, for the interactive numerical bifurcation analysis of ODE dynamical systems.

- ELLPACK [120] is a software for solving elliptic boundary value problems. It can use Hermite bicubic collocation methods for solving elliptic boundary value problems.

AUTO has probably been the most widely used continuation software package for ODE problems; its use has been cited in thousands of articles in the literature. Several other ODE BVP continuation software packages are based on AUTO, but AUTO remains, by far, the most efficient.

2.4 Using UML to present the object-oriented model

The complexity of software increases day by day. As a result, the necessity of having a rigorous modeling technique becomes more and more evident. Using a powerful modeling language standard is considered as one of essential factors for project success. *Model elements* (fundamental modeling concepts and semantics), *notation* (visual presentation of model elements), and *guidelines* (rules of usage within the context), are important elements of a modeling language, without which a modeling language cannot be considered as an acceptable one.

The *Unified Modeling Language* (*UML*) notation [104, 75, 151] is used increasingly in object-oriented modeling. The UML is a standard notation for specifying, visualizing, constructing, and documenting the object-oriented model. It is a set of successful skills for the modeling of large, complicated systems. Major parts of the UML language are constructed from the development and unification of *Booch* [20], *OMT* [123] (Object Modeling Technique) and *OOSE* (Object Oriented Software Engineering) notations. The UML notation has been widely accepted, and it is used as

modeling language in different domains. We have chosen this notation to present our model.

A fundamental idea behind the development of the UML has been to integrate the best practices in the industry, gathering widely varying views based on levels of abstraction, domains, architectures, life cycle stages, implementation technologies, *etc.* [110]. The UML has a well-designed architectural structure. Model elements are organized by packages. In each package (using the UML *class diagram*, *Object Constraint Language* expression and precise text), model element are explained in terms of abstract syntax, well-formedness rules , and semantics. Using the UML expressive visual modeling language, users can understand and exchange their models easily. Extensibility and specialization mechanisms of the UML permit extending the core concepts and using it in a new situation. Thus, for every specific domain, it is expected that the UML can be adapted as new requirements emerge. The UML is independent of any programming language and development procedures, and it can support every programming language. The UML provides a standard basis for understanding the modeling language. The UML also has higher-level development concepts such as collaborations, frameworks, patterns, and components. The UML can also be used for the modeling of concurrent and distributed systems. The UML has several models for illustrating a system:

- The Class model shows the static structure of the system,
- The state model represents the dynamic behavior of objects inside the system,
- The use case model specifies requirements of the user,
- The interaction model describes the scenarios and messages flows,
- The implementation model shows working units,
- The deployment model describes details related to the process allocation.

The UML has nine different diagrams. They are graphical notations that present different views of UML models. They are:

- Class diagrams (the static structure of the system in terms of classes and relationships),

- Sequence diagrams (the temporal representation of objects and their interactions),
- Collaboration diagrams (the spatial representation of objects, links and interactions),
- Object diagrams (objects and their relationships),
- Statechart diagrams (the behavior of a class in terms of states),
- Activity diagrams (the behavior of an operation as a set of actions),
- Use case diagrams (functions of a system from the user's point of view),
- Component diagrams (physical components of an application),
- Deployment diagrams (the deployment of components on particular pieces of hardware).

In this thesis, we have selected the UML notation as our object-oriented modeling language for the development of our PDE bifurcation software. Rational Rose software is a powerful tool that can be used for object-oriented modeling with the UML [109]. Unfortunately, Rational rose is a commercial software. Therefore we used the open source software ArgoUML [107]. Using the UML language, we present our prototype software structure, as well as an activity diagram and a statechart diagram of our bifurcation analysis of systems of nonlinear PDEs. This simplifies understanding the behavior of our system.

Chapter 3

The Finite Element Collocation Method for Linear ODE BVP

3.1 Introduction

As mentioned in Section 1.2, one of the objectives of this thesis, is the bifurcation analysis of PDE systems. We have chosen to use the finite element collocation method with piecewise polynomials to reach this goal. This method can also be used for the solution of ordinary differential equations (ODEs). In fact, one of advantages of this method is its generality; it can be used for a wide range of problems, and represents a type of unified method. In this chapter, as a first step toward the finite element collocation method for PDEs, we discuss finite element collocation methods with piecewise polynomials for BVP in second order ODEs. This is the method introduced originally by Russell and Shampine [126]. However, our presentation of the method is different, in order to prepare the reader for the technically more complicated case of the discontinuous piecewise polynomial collocation method for PDEs. The main characteristics of the ODE collocation method are:

- high order of accuracy is possible by increasing the number of collocation points,
- the piecewise polynomial solution is globally continuous, in fact smooth, for the case of second order ODEs. (However it is discontinous for the case of PDEs.)
- the linear systems that arise after discretisation can be solved efficiently by the method of nested dissection (This fact is not important for ODEs, but it will be important for PDEs, as it reduces the computational complexity),
- there are known mesh selection strategies.

The method gives us the high order accuracy that is typically required in numerical bifurcation studies. For this reason a subclass of the method has been used in COLSYS[11] and AUTO [48]. To have a robust solution procedure for solving the final equations, we have chosen to use a direct method. The tree structure of our domain decomposition is suitable for the use of the method of nested dissection. Using the nested dissection method for the case of ODEs does not reduce the computational complexity, as it does for PDEs. However, as will be see later, this method has advantages for bifurcation analysis, for ODEs as well as PDEs. Moreover, using the nested dissection method for ODEs will highlight the remarkable unity of the collocation method for ODEs and PDEs. For linear ODEs the collocation method can be defined equivalently as a type of generalized finite difference method. This approach will be described in detail in this chapter. We also give a local basis construction, which reduces computational cost for uniform and semi-uniform meshes. We will see that the finite difference formulation leads in a very natural way to the nested dissection procedure for solving the discretized equations.

3.2 Definition of the problem

As a model ODE problem, we consider the following second order linear ODE:

$$Lu = u''(x) + a(x)u'(x) + b(x)u(x) = f(x), \qquad 0 \leq x \leq 1. \tag{3.1}$$

where $a(x), b(x), f(x), u(x) \in R$ are sufficiently smooth functions. As boundary condition, we take

$$u(0) = u(1) = 0. \tag{3.2}$$

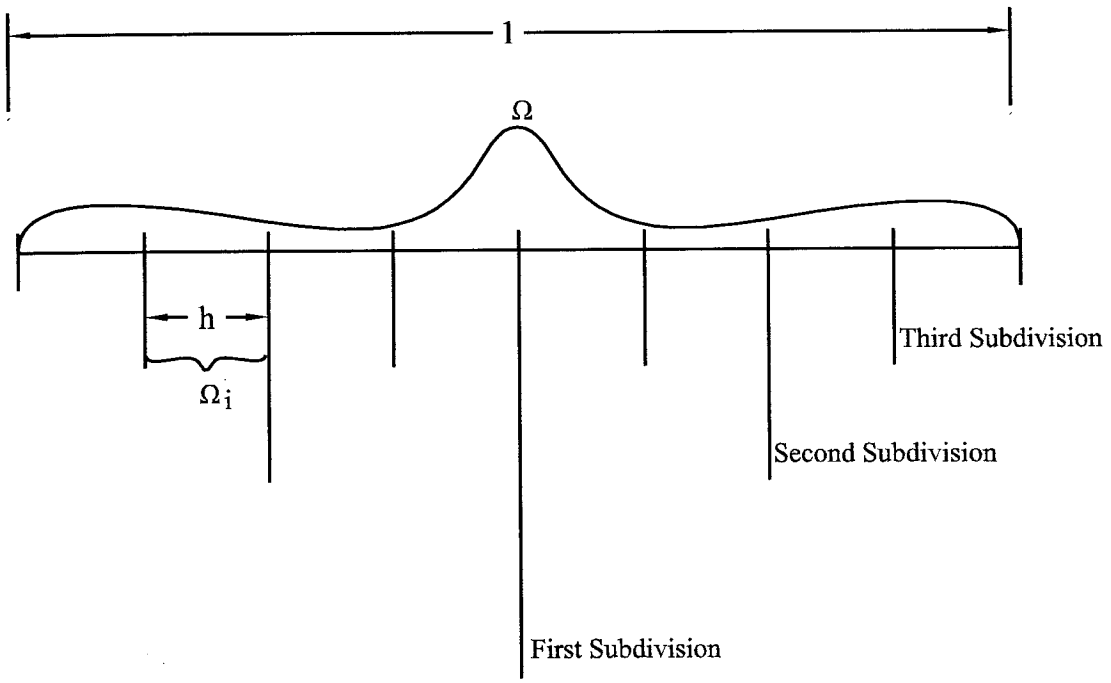


Figure 3.1: The recursive subdivision and the finite difference mesh

Although the procedure discussed in this chapter is for simple boundary conditions, it is also possible to treat this equation with more general boundary conditions.

3.3 Mesh generation

Consider the domain Ω , for simplicity a line segment $0 \leq x \leq 1$ in Figure 3.1. A binary tree data structure can keep the characteristics of our geometrical domain and help to have an efficient solution procedure, namely, a nested dissection method. The mesh generation is done as follows: initially the domain is subdivided into two subdomains or *regions*. Each subdomain may be subdivided into two smaller subdomains. This procedure is continued recursively until a desired level of refinement is achieved. The smallest regions are *finite elements*. These elements correspond to leaf nodes in the binary recursion tree. The recursion tree is not really necessary for the mesh generation, but this type of data structure is useful in the nested dissection algorithm and the local mesh refinement procedure. It is important to note that the final mesh need not be uniform, *i.e.*, the sizes of the mesh intervals need not be the same.

3.4 The finite difference formulation

Here we look at the finite difference formulation before the presentation of the equivalent collocation formulation. The finite difference formulation is especially useful in the nested dissection procedure. Consider any finite element Ω_i (Figure 3.2). Formally, if $u(x)$ is known on $\delta\Omega_i$ (boundary points of element Ω_i) and if Ω_i is sufficiently small then, under certain mild conditions, the solution $u(x)$ is defined in Ω_i. Equations (3.1) can be written as the following first order system:

$$\begin{aligned} u'(x) &= v(x), \\ v'(x) &= f(x) - a(x)v(x) - b(x)u(x), \end{aligned} \tag{3.3}$$

where $0 \leq x \leq 1$. As a finite difference approximation, one can use a mid-point finite difference formulation, the *Box scheme* of Keller [84], as follows:

$$\begin{aligned} &(u_i - u_{i-1})/h_i = \tfrac{1}{2}(v_i + v_{i-1}), \\ &(v_i - v_{i-1})/h_i = \tfrac{1}{2}(f_i + f_{i-1}) - \tfrac{1}{2}a_{i-1/2}(v_i + v_{i-1}) - \tfrac{1}{2}b_{i-1/2}(u_i + u_{i-1}), \\ &\qquad\qquad i = 1, 2, \cdots, N, \\ &u_0 = u_N = 0. \end{aligned} \tag{3.4}$$

Here N is the number of finite elements and u_i denotes the approximate solution on a boundary point of the finite element, that is $u_i \approx u(x_i)$. Similarly, v_i denotes the derivative of u at x_i, *i.e.*, $v_i \approx u'(x_i)$, and h_i is equal to $\mid x_i - x_{i-1} \mid$. The points x_i, $i = 1, \cdots, N-1$, will be identified in Section 3.8 as *matching points*. Using Taylor expansions, one can show that this approximation is of order $O(h_i^2)$, (*cf.* [12]). We can solve above Equations (3.4) for v_i and v_{i-1} in terms of u_i, u_{i-1}, f_i and f_{i-1}

$$\begin{aligned} v_{i-1} &= (\frac{b_{i-1/2}h_i - 2a_{i-1/2}}{4} - \frac{1}{h_i})u_{i-1} + (\frac{b_{i-1/2}h_i + 2a_{i-1/2}}{4} + \frac{1}{h_i})u_i - \frac{h_i}{4}f_i - \frac{h_i}{4}f_{i-1}, \\ v_i &= (\frac{2a_{i-1/2} - b_{i-1/2}h_i}{4} - \frac{1}{h_i})u_{i-1} - (\frac{b_{i-1/2}h_i + 2a_{i-1/2}}{4} - \frac{1}{h_i})u_i + \frac{h_i}{4}f_i + \frac{h_i}{4}f_{i-1}, \\ &\qquad\qquad i = 1, 2, \cdots, N, \\ u_0 &= u_N = 0. \end{aligned}$$

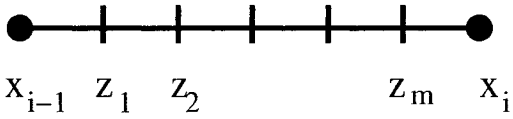


Figure 3.2: A finite element of Ω

More generally, for any finite element Ω_i, one can consider the following discretisation of Equation (3.1):

$$\begin{aligned} v_{i-1} &= \alpha^i_{11} u_{i-1} + \alpha^i_{12} u_i + \textstyle\sum_{j=1}^m \beta^i_{1j} f(z_{ij}), \\ v_i &= \alpha^i_{21} u_{i-1} + \alpha^i_{22} u_i + \textstyle\sum_{j=1}^m \beta^i_{2j} f(z_{ij}), \qquad i = 1, \cdots, N, \end{aligned} \tag{3.5}$$

where m is the number of evaluation points z_{ij} inside Ω_i, which are identified in Section 3.7 as *collocation points*. (From now on, we drop the indices i, for simplicity of notation.) Thus, for each finite element of Ω there is a discrete equation corresponding to each of the two boundary points of the finite element. The coefficients $\alpha_{11}, \alpha_{12}, \alpha_{21}, \alpha_{21}, \beta_{1j}$ and β_{2j} in Equations (3.5) can be determined by requiring Equations (3.5) to be satisfied exactly for a basic set of equations with known solutions. These test problems are of the form of Equation (3.1), with $f = L\phi$, where ϕ is a polynomial, for which an obvious solution is $u = \phi$. More precisely, let P_{m+2} be the space of polynomials of degree $m+1$ (order $m+2$) or less, $P_{m+2} = Span\{\phi_1, \cdots, \phi_{m+2}\}$. Then we require that

$$\begin{aligned} \phi'(x_{i-1}) &= \alpha_{11}\phi(x_{i-1}) + \alpha_{12}\phi(x_i) + \textstyle\sum_{j=1}^m \beta_{1j} L\phi(z_j), \\ \phi'(x_i) &= \alpha_{21}\phi(x_{i-1}) + \alpha_{12}\phi(x_i) + \textstyle\sum_{j=1}^m \beta_{2j} L\phi(z_j), \end{aligned} \tag{3.6}$$

for all $\phi \in P_{m+2}$. Let

$$u = (u_{i-1}, u_i)^T,$$

$$v = (v_{i-1}, v_i)^T,$$

$$f = (f(z_1), \cdots, f(z_m))^T,$$

and take

$$A = \begin{pmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{pmatrix}, \qquad B = \begin{pmatrix} \beta_{11} & \cdots & \beta_{1m} \\ \beta_{21} & \cdots & \beta_{2m} \end{pmatrix}. \tag{3.7}$$

Then Equation (3.5) can be written as

$$v = Au + Bf. \tag{3.8}$$

If we define

$$\Phi = \begin{pmatrix} \phi_1(x_{i-1}) & \phi_1(x_i) \\ \vdots & \vdots \\ \phi_{m+2}(x_{i-1}) & \phi_{m+2}(x_i) \end{pmatrix}, \qquad L_\Phi = \begin{pmatrix} L\phi_1(z_1) & \cdots & L\phi_1(z_m) \\ \vdots & & \vdots \\ L\phi_{m+2}(z_1) & \cdots & L\phi_{m+2}(z_m) \end{pmatrix}, \quad (3.9)$$

and

$$R_\Phi = \begin{pmatrix} \phi_1'(x_{i-1}) & \phi_1'(x_i) \\ \vdots & \vdots \\ \phi_{m+2}'(x_{i-1}) & \phi_{m+2}'(x_i) \end{pmatrix}, \quad (3.10)$$

then Equations (3.6) can be written as

$$(\Phi \mid L_\Phi) \begin{pmatrix} A^T \\ B^T \end{pmatrix} = R_\Phi. \quad (3.11)$$

For the finite difference approximation to be well defined, the matrix $(\Phi \mid L_\Phi)$ must be nonsingular. (A related condition is analyzed in [40].) Although this condition is not automatically satisfied, there are many possible schemes for choosing the points z_j so that this condition is satisfied. One such scheme is given in Section 3.6, see Theorem 3.6.1. Note that if the coefficient functions $a(x)$ and $b(x)$ in Equation (3.1) are constant and if all finite elements are identical then Equation (3.11) only needs to be solved only once. This observation is further developed in Section 3.11.1. Of course, in addition to the finite difference Equations (3.5), the boundary conditions must be satisfied, *i.e.*,

$$u_0 = u_N = 0. \quad (3.12)$$

3.5 Example

Consider the simple linear operator $L = d/dx^2$. We take a line segment element as in Figure 3.3. For simplicity assume that it is centered at $x = 0$ and has length h. Let $m = 1$ and take the collocation point z_1 to be the center of the segment. Corresponding to the matching points $x_{-h/2}, x_{h/2}$ the finite difference approximation, Equation (3.5), takes the form

$$\begin{aligned} v_{-h/2} &= \alpha_{11} u_{-h/2} + \alpha_{12} u_{h/2} + \beta_1 f(z_1), \\ v_{h/2} &= \alpha_{21} u_{-h/2} + \alpha_{22} u_{h/2} + \beta_2 f(z_1). \end{aligned} \quad (3.13)$$

As basis of $P_{2+m} = P_3$ we choose polynomials $\{1, x, x^2\}$. The coefficients α_{ij} and β_i

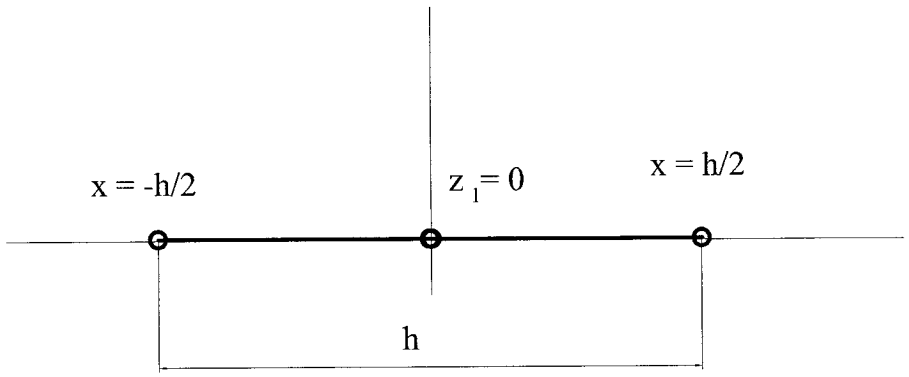


Figure 3.3: A 1D element

are obtained by solving Equation (3.11), which here becomes

$$\begin{pmatrix} 1 & 1 & 0 \\ -h/2 & h/2 & 0 \\ h^2/4 & h^2/4 & 2 \end{pmatrix} \begin{pmatrix} \alpha_{11} & \alpha_{21} \\ \alpha_{12} & \alpha_{22} \\ \beta_1 & \beta_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ -h & h \end{pmatrix}.$$

Upon solving this system one finds that "finite difference" equations are given by

$$v_{-h/2} = (u_{h/2} - u_{-h/2})/h - hf(z_1)/2,$$
$$v_{h/2} = (u_{h/2} - u_{-h/2})/h + hf(z_1)/2.$$

3.6 Lagrange basis functions

The system (3.11) must be solved for coefficients α_{ij} and β_{ij}, for each finite element. In this section, we show how these coefficients can be computed using Lagrange basis functions. We also investigate the singularity of the system (3.11).

We divide the basis of P_{m+2} into two groups, namely $\{\phi_1, \phi_2\}$ and $\{\psi_1, \cdots, \psi_m\}$. For ϕ_i, $i = 1, 2$ and ψ_j, $j = 1, \cdots, m$, we use a Lagrange basis corresponding to the set of points $\{x_{i-1}, x_i\} \bigcup \{z_1, \cdots, z_n\}$ such that:

$$\phi_1 = 1 \quad \text{at } x_{i-1}, \quad \phi_1 = 0 \quad \text{at } \{x_i, z_1, \cdots, z_m\},$$
$$\phi_2 = 1 \quad \text{at } x_i, \quad \phi_2 = 0 \quad \text{at } \{x_{i-1}, z_1, \cdots, z_m\},$$

and

$$\psi_1 = 1 \quad \text{at } z_1, \quad \psi_1 = 0 \quad \text{at } \{x_{i-1}, x_i, z_2, \cdots, z_m\},$$
$$\psi_2 = 1 \quad \text{at } z_2, \quad \psi_2 = 0 \quad \text{at } \{x_{i-1}, x_i, z_1, z_3, \cdots, z_m\},$$
$$\cdots$$

$$\psi_m = 1 \quad \text{at } z_m, \quad \psi_m = 0 \quad \text{at } \{x_{i-1}, x_i, z_1, \cdots, z_{m-1}\}.$$

Then the matrices Φ, L_Φ and R_Φ in the preceding section can be rewritten as

$$\Phi = \begin{pmatrix} \tilde{\Phi} \\ \Psi \end{pmatrix} = \begin{pmatrix} \phi_1(x_{i-1}) & \phi_1(x_i) \\ \phi_2(x_{i-1}) & \phi_2(x_i) \\ \psi_1(x_{i-1}) & \psi_1(x_i) \\ \vdots & \vdots \\ \psi_m(x_{i-1}) & \psi_m(x_i) \end{pmatrix}, \tag{3.14}$$

$$L_\Phi = \begin{pmatrix} L_{\tilde{\Phi}} \\ L_\Psi \end{pmatrix} = \begin{pmatrix} L\phi_1(z_1) & \cdots & L\phi_1(z_m) \\ L\phi_2(z_1) & \cdots & L\phi_2(z_m) \\ L\psi_1(z_1) & \cdots & L\psi_1(z_m) \\ \vdots & & \vdots \\ L\psi_m(z_1) & \cdots & L\psi_m(z_m) \end{pmatrix}, \tag{3.15}$$

and

$$R_\Phi = \begin{pmatrix} R_{\tilde{\Phi}} \\ R_\Psi \end{pmatrix} = \begin{pmatrix} \phi_1'(x_{i-1}) & \phi_1'(x_i) \\ \phi_2'(x_{i-1}) & \phi_2'(x_i) \\ \psi_1'(x_{i-1}) & \psi_1'(x_i) \\ \vdots & \vdots \\ \psi_m'(x_{i-1}) & \psi_m'(x_i) \end{pmatrix}. \tag{3.16}$$

As a result, Equation (3.11) now becomes

$$\begin{pmatrix} \tilde{\Phi} & L_{\tilde{\Phi}} \\ \Psi & L_\Psi \end{pmatrix} \begin{pmatrix} A^T \\ B^T \end{pmatrix} = \begin{pmatrix} R_{\tilde{\Phi}} \\ R_\Psi \end{pmatrix}. \tag{3.17}$$

Evaluating $\tilde{\Phi}$ and Ψ, this system becomes

$$\begin{pmatrix} I & L_{\tilde{\Phi}} \\ O & L_\Psi \end{pmatrix} \begin{pmatrix} A^T \\ B^T \end{pmatrix} = \begin{pmatrix} R_{\tilde{\Phi}} \\ R_\Psi \end{pmatrix}, \tag{3.18}$$

which can be solved efficiently in two stages, namely

$$L_\Psi B^T = R_\Psi, \tag{3.19}$$

$$A^T = R_{\tilde{\Phi}} - L_{\tilde{\Phi}} B^T. \tag{3.20}$$

It is clear that Equation (3.11) is nonsingular if L_Ψ in Equation (3.19) is nonsingular.

Theorem: 3.6.1 *If $h_i > 0$ is sufficiently small and the points z_i are distinct and chosen according to a template then the above finite difference construction is well defined. (i.e., L_Ψ is nonsingular)*

PROOF.

If we take

$$L^2u(x) = u''(x),$$

$$L^1u(x) = a(x)u'(x),$$

and

$$L^0u(x) = b(x)u(x),$$

then we have

$$Lu = L^2u + L^1u + L^0u.$$

L_Ψ in Equation (3.15) can be written as:

$$L_\Psi = L^2_\Psi + L^1_\Psi + L^0_\Psi, \tag{3.21}$$

where

$$L^0_\Psi = \begin{pmatrix} b(z_1)\psi_1(z_1) & \cdots & b(z_m)\psi_1(z_m) \\ b(z_1)\psi_2(z_1) & \cdots & b(z_m)\psi_2(z_m) \\ \vdots & & \vdots \\ b(z_1)\psi_m(z_1) & \cdots & b(z_m)\psi_m(z_m) \end{pmatrix}, \tag{3.22}$$

$$L^1_\Psi = \begin{pmatrix} a(z_1)\psi_1'(z_1) & \cdots & a(z_m)\psi_1'(z_m) \\ a(z_1)\psi_2'(z_1) & \cdots & a(z_m)\psi_2'(z_m) \\ \vdots & & \vdots \\ a(z_1)\psi_m'(z_1) & \cdots & a(z_m)\psi_m'(z_m) \end{pmatrix}, \tag{3.23}$$

$$L^2_\Psi = \begin{pmatrix} \psi_1''(z_1) & \cdots & \psi_1''(z_m) \\ \psi_2''(z_1) & \cdots & \psi_2''(z_m) \\ \vdots & & \vdots \\ \psi_m''(z_1) & \cdots & \psi_m''(z_m) \end{pmatrix}. \tag{3.24}$$

First we prove that L^2_Ψ is not singular. If, on the contrary, L^2_Ψ is singular, then we can find a vector $c \neq 0$ such that

$${L^2_\Phi}^T c = \begin{pmatrix} \psi_1''(z_1) & \cdots & \psi_m''(z_1) \\ \vdots & & \vdots \\ \psi_1''(z_m) & \cdots & \psi_m''(z_m) \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_m \end{pmatrix} = 0. \tag{3.25}$$

If we define $q(x) = \sum_{k=1}^m c_k\psi_k(x)$ then, using the above equation, we have

$$q''(z_j) = 0, \quad j = 1, \cdots, m. \tag{3.26}$$

However, $q \in P_{m+2}$, and $q'' \in P_m$ (space of polynomials of degree m-1) hence we must have that $q'' \equiv 0$ everywhere inside the element. On the other hand, all of the ψ_k, $k =$

$1, \cdots, m$, are zero at the boundaries of the element, *i.e.*, we have $q(x_{i-1}) = q(x_i) = 0$. As a result, we conclude that $q \equiv 0$ inside the element. Therefore the vector c must be the zero vector which contradicts the assumption. Thus L^2_Ψ cannot be singular.

Equation (3.21) can be written as:

$$L_\Psi = L^2_\Psi(I + [L^2_\Psi]^{-1}(L^1_\Psi + L^0_\Psi)). \tag{3.27}$$

Each term of L^0_Ψ is $b(x)$ multiplied by a Lagrange basis function of order $m+1$ (Lagrange basis for two end points plus $m-1$ points z_i). If we consider x_{i-1} as z_0 and x_i as z_{m+1}, then a typical Lagrange basis can be written as

$$\Psi_i(x) = \frac{(x-z_0)\cdots(x-z_{i-1})(x-z_{i+1})\cdots(x-z_{m+1})}{(z_i-z_0)\cdots(z_i-z_{i-1})(z_i-z_{i+1})\cdots(z_i-z_{m+1})}. \tag{3.28}$$

For an element of size h, each term of numerator $|x - z_k|$ is smaller than h and if we write the smallest factor of denominator (smallest $|z_i - z_k|$) as rh, where r is a constant $(0 < r < 1)$, we have

$$|\Psi_i(x)| \leq h^{m+1}/(rh)^{m+1} = r^{-(m+1)} \quad \text{(a constant)}.$$

By assumption, $b(x)$ and $a(x)$ are bounded. As a result, in Equation (3.27), $\| L^0_\Psi \|_\infty$ is of order $\mathcal{O}(1)$. If we take the first and the second derivative of Ψ_i in Equation (3.28) we have

$$\Psi_i'(x) = \frac{(x-z_1)\cdots(x-z_{m+1}) + \cdots + (x-z_0)\cdots(x-z_m)}{(z_i-z_0)\cdots(z_i-z_{i-1})(z_i-z_{i+1})\cdots(z_i-z_{m+1})},$$

or

$$\Psi_i'(x) = \frac{\sum_{j=0\ j\neq i}^{m+1} \prod_{k=0\ k\neq i,\ k\neq j}^{m+1}(x-z_k)}{\prod_{k=0\ k\neq i}^{m+1}(z_i-z_k)},$$

and

$$\Psi_i''(x) = \frac{\sum_{j=0\ j\neq i}^{m+1}\sum_{l=0\ l\neq i,\ l\neq j}^{m+1}\prod_{k=0\ k\neq i,\ k\neq j,\ k\neq l}^{m+1}(x-z_k)}{\prod_{k=0\ k\neq i}^{m+1}(z_i-z_k)}.$$

As a result

$$|\Psi_i'(x)| \leq (m+1)h^m/(rh)^{m+1} = (m+1)r^{-(m+1)}/h,$$

and

$$|\Psi_i''(x)| \leq m(m+1)h^{m-1}/(rh)^{m+1} = m(m+1)r^{-(m+1)}/h^2.$$

Therefore $\| L^1_\Psi \|_\infty$ is of order $\mathcal{O}(1/h)$, and $\| L^2_\Psi \|_\infty$ is of order $\mathcal{O}(1/h^2)$. As L^2_Ψ is not singular we have that $\| [L^2_\Psi]^{-1}(L^1_\Psi + L^0_\Psi) \|_\infty$ is of order $\mathcal{O}(h)$. As a result, if we

chose h sufficiently small then $\|[L_\Psi^2]^{-1}(L_\Psi^1 + L_\Psi^0)\|_\infty < 1$. Using *Banach lemma* [46], we conclude that $I + [L_\Psi^2]^{-1}(L_\Psi^1 + L_\Psi^0)$ is not singular. Therefore, L_Ψ in Equation (3.27) is not singular.

3.7 The collocation formulation

If $(\Phi \mid L_\Phi)$ in Equation (3.11) is nonsingular then the finite difference scheme is equivalent to a collocation scheme. To show this, we construct a collocation formulation as follows:

- First, we associate a polynomial $p(x) \in P_{m+2}$ (space of polynomials of order $m+2$) to each finite element.

- Second, for any two adjacent finite elements, we require that at the point x_i on the common boundary we have :

  $\triangleright$ the same values of the neighboring polynomials,
  and
  $\triangleright$ the same values of the derivatives of the neighboring polynomials. (3.29)

- Third, each polynomial must satisfy the *collocation* equations

$$Lp(z_k) = f(z_k), \qquad k = 1, \cdots, m, \tag{3.30}$$

  at all collocation points z_k of the corresponding finite element.

- Finally the boundary conditions must be satisfied, *i.e.*, for each boundary adjacent element we have

$$p(x) = 0 \text{ at the points } x = 0 \text{ and } x = 1. \tag{3.31}$$

The precise meaning of *equivalence* of the finite difference scheme and the collocation scheme is given in the following theorem.

Theorem: 3.7.1 *For each finite element Ω_i let the matrix $(\Phi \mid L_\Phi)$ be nonsingular and let the matrices A and B be defined by Equation (3.11). Suppose we have a solution of the collocation scheme, Equations (3.29-3.31). If for each finite element the local polynomial is evaluated at the points x_i then the resulting values satisfy the*

finite difference Equation (3.8). Conversely, suppose we have a solution to the coupled system of all finite difference Equations, i.e., Equation (3.8) and the boundary conditions Equation (3.12). Then, locally, for each element, we can interpolate the finite difference solution by a polynomial $p(x) \in P_{m+2}$ that satisfies the differential equation at the collocation points. These polynomials exist, and collectively they satisfy the continuity properties in (3.29).

PROOF.

Take $p(x) = \sum_{i=1}^{m+2} c_i \phi_i(x)$, where $Span\{\phi_1, \cdots, \phi_{m+2}\} = P_{m+2}$. Then the collocation Equations (3.30) can be written as $\sum_{i=1}^{m+2} c_i L\phi_i(z_k) = f(z_k)$. Using the definitions of L_Φ, Φ and R_Φ, Equations (3.9) and (3.10), we can write

$$L_\Phi^T c = f,$$

where $c = (c_1, \cdots, c_{m+2})^T$, and

$$u = \Phi^T c, \;\; v = R_\Phi^T c.$$

Defining A and B as the solution of Equation (3.11), then the variables u and v satisfy the finite difference Equation (3.8), because

$$\begin{aligned} v - Au - Bf &= R_\Phi^T c - A\Phi^T c - Bf \\ &= [c^T(R_\Phi - \Phi A^T)]^T - Bf, \end{aligned}$$

and, using Equation (3.11),

$$\begin{aligned} v - Au - Bf &= [c^T(L_\Phi B^T)]^T - Bf \\ &= BL_\Phi^T c - Bf = B(L_\Phi^T c - f) = 0. \end{aligned}$$

Conversely let (u, v) be a solution of the finite difference Equation (3.8). Define c by $\begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix} c = \begin{pmatrix} u \\ f \end{pmatrix}$, and set $p(x) = \sum_{i=1}^{m+2} c_i \phi_i(x)$. Thus c satisfies $\Phi^T c = u$, and $L_\Phi^T c = f$ (*i.e.*, the collocation equations are satisfied). Neighboring polynomials are continuous at the matching points x_i, because $(p(x_{i-1}), p(x_i))^T = \Phi^T c = u$. The continuity of p' is also satisfied because

$$\begin{aligned} \begin{pmatrix} p(x_{i-1})' \\ p(x_i)' \end{pmatrix} = R_\Phi^T c &= (\Phi A^T + L_\Phi B^T)^T c \\ &= A\Phi^T c + BL_\Phi^T c \\ &= Au + Bf = v. \end{aligned}$$

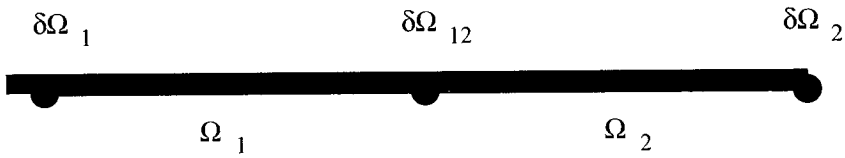


Figure 3.4: Two adjacent elements

3.8 Nested dissection

The nested dissection technique is valuable for PDEs; here we illustrate this method for ODEs in preparation for the PDE case. The nested dissection procedure consists of recursive elimination of unknowns u and v on boundaries separating adjacent regions. Here "adjacent" means two subdomains that result from the subdivision of the larger domain formed by their union, *i.e.*, these two regions correspond to descendant nodes of a common parent node in the binary tree. "Recursion" means that the elimination starts at the leaves and terminates at the root of the tree. This procedure results in the elimination of all interior unknowns. One is left with one equation for each x_i on $\delta\Omega$ (the boundary of the domain). Boundary conditions can then be used to determine the values of u and v at the x_i on $\delta\Omega$. Thereafter, a recursive back-substitution gives the values of u (and v) at each interior matching point (separators of adjacent regions).

3.8.1 Local elimination

Consider two arbitrary adjacent regions Ω_1 and Ω_2, as in Figure 3.4. It is not necessary that these regions be finite elements, *i.e.*, they can correspond to any neighboring nodes in the recursion tree. The elimination of the unknowns u and v on the common boundary is done in the opposite direction of the domain decomposition procedure *i.e.*, upwards in the binary tree. The common boundary of the two adjacent regions will be called $\delta\Omega_{12}$ (see Figure 3.4), and the remaining parts of the boundaries of Ω_1 and Ω_2 will be called $\delta\Omega_1$ and $\delta\Omega_2$ respectively. The vector u in the finite difference Equation (3.8) is split accordingly into u_{12} and u_1 for region Ω_1 and u_{21} $(= u_{12})$ and u_2 for region Ω_2. The vector v is similarly split into two parts. Thus for region Ω_1, Equations (3.8) can be written in the form

$$v_1 \quad = A^1_{11}u_1 + A^1_{12}u_{12} + g_1, \tag{3.32}$$

$$v_{12} = A^1_{21}u_1 + A^1_{22}u_{12} + g_{12}, \tag{3.33}$$

while for Ω_2 they have the form

$$v_2 = A^2_{11}u_2 + A^2_{12}u_{21} + g_2, \tag{3.34}$$

$$v_{21} = A^2_{21}u_2 + A^2_{22}u_{21} + g_{21}. \tag{3.35}$$

The superscript represent the region number. Above $\begin{pmatrix} g_1 \\ g_{12} \end{pmatrix}$ and $\begin{pmatrix} g_2 \\ g_{21} \end{pmatrix}$ correspond to the term Bf in Equation (3.8) for region Ω_1 and Ω_2 respectively. From the continuity relations (3.29) we have

$$u_{12} = u_{21}, \tag{3.36}$$

$$v_{12} = v_{21}.$$

From Equations (3.33), (3.35), and (3.36) it follows that

$$u_{12} = -E^{-1}(A^1_{21}u_1 - A^2_{21}u_2 + g_{12} - g_{21}), \tag{3.37}$$

where E is defined as:

$$E \equiv A^1_{22} - A^2_{22}. \tag{3.38}$$

Substituting (3.38) into (3.32) and (3.34), and again using (3.36), one obtains

$$\begin{aligned} v_1 &= A^1_{11}u_1 - A^1_{12}E^{-1}(A^1_{21}u_1 - A^2_{21}u_2 + g_{12} - g_{21}) + g_1, \\ v_2 &= A^2_{11}u_2 - A^2_{12}E^{-1}(A^1_{21}u_1 - A^2_{21}u_2 + g_{12} - g_{21}) + g_2, \end{aligned} \tag{3.39}$$

which is of the form

$$v_1 = C_{11}u_1 + C_{12}u_2 + \hat{g}_1, \tag{3.40}$$

$$v_2 = C_{21}u_1 + C_{22}u_2 + \hat{g}_2,$$

where

$$\begin{aligned} C_{11} &= A^1_{11} - A^1_{12}E^{-1}A^1_{21}, \\ C_{12} &= A^1_{12}E^{-1}A^2_{21}, \\ C_{21} &= -A^2_{12}E^{-1}A^1_{21} \\ C_{22} &= A^2_{11} + A^2_{12}E^{-1}A^2_{21}, \end{aligned}$$

and

$$\hat{g}_1 = g_1 - A_{12}^1 E^{-1}(g_{12} - g_{21}),$$
$$\hat{g}_2 = g_2 - A_{12}^2 E^{-1}(g_{12} - g_{21}).$$

Equation (3.40) represents the discrete equation for the enlarged region, that is, for the union of Ω_1 and Ω_2, after the elimination of common boundary unknowns u_{12} and v_{12}. These new equations are again of the same form as Equation (3.8). Therefore one can carry out the same procedure for the next higher level in the recursion tree, and again we obtain an equation of the form of Equation (3.40). Recursively, one can continue this procedure until the root of our binary tree.

While the nested dissection procedure for ODEs, as described above, has no advantage in terms of computational complexity, it is used in the AUTO software package [48, 52, 51, 56], since it allows to extract the so-called *Floquet multipliers*, which are important in a bifurcation analysis.

3.8.2 Complexity

For one-dimensional ($1D$) problems, *i.e.*, for ODEs, each step of the nested dissection procedure has the same number of operations. For our model problem, Equation (3.1), we have, in each step one subtraction and one division for evaluating E^{-1}, 8 multiplications, one addition and one subtraction for evaluating C_{11} to C_{22} and 4 subtractions and 4 multiplications for evaluating $\hat{g}_1$ and $\hat{g}_2$. As a result, we have a total of 20 basic mathematical operations. This number of operations does not depend on the position of the region in the binary tree and it is always the same. Therefore, the order of complexity of the nested dissection method for $1D$ problems only depends on the number of interior nodes of the binary tree. For a domain with N elements, we must construct a binary tree with $N-1$ interior nodes. As a result, the number of operations is $20 * (N-1)$. Thus, the order of nested dissection procedure for $1D$ problems is $\mathcal{O}(N)$.

3.9 Stability and convergence theory

In this section, we show that if the differential equation has a unique solution, then the collocation system has also a unique solution, for sufficiently small h, and

the discretized solution converges to the analytical solution as the mesh size tends to zero.

Much work has already been done on the stability and convergence theory of the collocation method in ODE BVP; for example, we can mention the paper by Russell and Shampine [126], the book by Ascher, Mattheij and Russell [12], and the work of de Boor and Swartz [35] who prove the superconvergence for the case of local Gauss collocation points, which has made collocation for BVP ODEs so powerful and widely used, *e.g.*, in AUTO [43, 51, 23, 48, 52]. In this section, we use a different technique for proving stability and convergence of the collocation method for ODE BVP, adapted from a paper by Doedel [42], which may also be applicable to the case of PDEs. Certain stability results for some modified versions of our methods for PDEs are reported in the thesis by Goldlücke [73]; however, the precise methods considered there do not quite correspond to the ones considered in this thesis.

Consider mesh points $\{x_0, x_1, \cdots, x_N\}$ over a domain $[0,1]$ with $x_0 = 0,\ x_N = 1$. Define

$$h_i = x_i - x_{i-1}, \quad H = \{h_1, h_2, \cdots, h_N\}, \quad \mid h \mid = \max_{1 \le i \le N} h_i.$$

Define the following space of piecewise polynomials over the domain:

$$\mathbf{P_k^h} \equiv \{p_h : p_h \in C^1[0,1],\ p_h|_{[x_{i-1},x_i]} \equiv p_i \in P_k,\ p_h(0) = p_h(1) = 0\},$$

meaning, each member p_h of the space has a C^1 continuity over the domain, and each segment of p_h, for example, p_i defined over the interval $[x_{i-1}, x_i]$ belongs to the space of polynomials of degree $k-1$ (order k), *i.e.*, P_k. The boundary conditions must also be satisfied, *i.e.*, $p_h(0) = p_h(1) = 0$. Also define

$$\| p_h \|_m \equiv \max_{1 \le i \le N} \max_{[x_{i-1},x_i]} \max_{0 \le j \le m} \mid p_i^{(j)} \mid, \qquad 0 \le m \le k-1,$$

where $p_i^{(j)}$ is the j^{th} derivative of p_i.

Lemma: 3.9.1 *Let $\{h^\nu\}_{\nu=1}^\infty$ be a sequence of meshes with $\mid h \mid \to 0$ as $\nu \to \infty$, with corresponding $\{\mathbf{p_{h^\nu}}\}_{\nu=1}^\infty$, $\mathbf{p_{h^\nu}} \in \mathbf{P_k^{h^\nu}}$ and $\| \mathbf{p_{h^\nu}} \|_1$ is bounded. Then there exists a subsequence of $\{\mathbf{p_{h^\nu}}\}_{\nu=1}^\infty$, also denoted by $\{\mathbf{p_{h^\nu}}\}_{\nu=1}^\infty$, and a function $P \in C^1[0,1]$ such that:*

$$\| \mathbf{p_{h^\nu}} - P \|_1 \to 0 \quad \text{as} \quad \nu \to \infty.$$

Proof: This follows from the Arzela-Ascoli theorem [95, 99].

Theorem: 3.9.1 *Assume that the following ODE BVP has only* $u \equiv 0$ *as a solution.*

$$Lu = 0, \quad \textit{with boundary conditions} \quad u(0) = 0, \; u(1) = 0,$$

with L *as in Equation (3.1), and* $a,\ b,\ f \in C[0,\ 1]$. *Assume that for each element the collocation points are placed according to a fixed template. Then for all meshes* h^ν, *with* $\mid h^\nu \mid$ *sufficiently small, the collocation equations*

$$Lp_i^\nu(z_{ij}) = f(z_{ij}), \qquad i = 1, \cdots, N, \qquad j = 1, \cdots, m, \qquad p_i^\nu \in P_{m+2},$$

$$\mathbf{p_{h^\nu}}(0) = \mathbf{p_{h^\nu}}(1) = 0, \qquad \mathbf{p_{h^\nu}} \in \mathbf{P_{m+2}^{h^\nu}}, \qquad \mathbf{p_{h^\nu}}|_{[x_{i-1}, x_i]} \equiv p_i^\nu,$$

where p_i^ν *is a segment of the polynomial* $\mathbf{p_{h^\nu}}$ *over the element* i,

- *first: has a unique solution* $\mathbf{p_{h^\nu}}$,
- *second:* $\| \mathbf{p_{h^\nu}} \|_1 \leq K \| f \|_\infty$, *where* K *does not depends on* h.

Proof of the first part: The collocation equations constitute a linear system of N*(m+2) equations and N*(m+2) unknown coefficients; namely,

- N*m collocation equations
- 2(N-1) matching conditions between adjacent elements
- 2 boundary conditions.

If, in contradiction to the first part of the theorem, we do not have a unique solution for all small $\mid h^\nu \mid$, then there must be a sequence of meshes $\{h^\nu\}_{\nu=1}^\infty$, with $\mid h^\nu \mid \to 0$ as $\nu \to \infty$, for which there exists a sequence of piecewise polynomials $\{\mathbf{p_{h^\nu}}\}_{\nu=1}^\infty$ such that $\| \mathbf{p_{h^\nu}} \|_1 = 1$, that satisfies collocation equations

$$Lp_i^\nu(z_{ij}) = 0, \quad i = 1, \cdots, N, \quad j = 1, \cdots, m, \quad p_1^\nu(0) = p_N^\nu(1) = 0.$$

For each mesh h^ν we have

$$f_{ij}^\nu \equiv f^\nu(z_{ij}) \equiv 0 \text{ and } \| f^\nu \|_\infty = 0.$$

Since $p_i^\nu \in P_{m+2}$ we have $p_i^{\nu''} \in P_m$. Thus $p_i^{\nu''}(x)$ is completely determined by $\{p_i^{\nu''}(z_{ij}) \equiv p_{ij}^{\nu''}\}_{j=1}^m$, and therefore we can write $p_i^{\nu''}$ in terms of Lagrange basis functions for $l_{ij} \in P_m$, $l_{ij}(z_{ij}) = \delta_{ij}$

$$p_i^{\nu''}(x) = \sum_{j=1}^{m} l_{ij}(x) p_{ij}^{\nu''}.$$

As the collocation points (z_{ij}) are chosen according to a template, the $l_{ij}(x)$ are bounded, *i.e.*, $l_{ij}(x) \leq K_l$ (a constant), and we can write

$$\begin{aligned} \max_{x\in[x_{i-1},x_i]} \mid p_i^{\nu''}(x) \mid &= \max \mid \sum_{j=1}^{m} l_{ij}(x) p_{ij}^{\nu''} \mid \\ &\leq mK_l \max_i \mid f^\nu(z_{ij}) - a(z_{ij}) p_i^{\nu'}(z_{ij}) - b(z_{ij}) p_i^\nu(z_{ij}) \mid . \end{aligned} \tag{3.41}$$

Since $p_i^{\nu'}(z_{ij})$ and $p_i^\nu(z_{ij})$ are bounded quantities, and $f^\nu(z_{ij}) \equiv 0$, we have

$$\max_{x\in[x_{i-1},x_i]} \mid p_i^{\nu''}(x) \mid \leq K \text{ (a constant)}.$$

From Lemma (3.9.1) we conclude that there exist a subsequence of $\{\mathbf{p_h}^\nu\}_{\nu=1}^\infty$ and a function $p \in C^1[0,1]$ such that:

$$\| \mathbf{p_h}^\nu - p \|_1 \to 0 \quad \text{as} \quad \nu \to \infty.$$

Choose an arbitrary point $s \in [0,1]$ inside an element i, *i.e.*, $s \in [x_{i-1}, x_i]$. We can write

$$\begin{aligned} &p_i^{\nu''}(s) + a(s)p_i^{\nu'}(s) + b(s)p_i^\nu(s) = \textstyle\sum_{j=1}^m l_{ij}(s) p_{ij}^{\nu''} + a(s)p_i^{\nu'}(s) + b(s)p_i^\nu(s) = \\ &\textstyle\sum_{j=1}^m l_{ij}(s)\{f^\nu(z_{ij}) - a(z_{ij})p_i^{\nu'}(z_{ij}) - b(z_{ij})p_i^\nu(z_{ij})\} + a(s)p_i^{\nu'}(s) + b(s)p_i^\nu(s), \end{aligned} \tag{3.42}$$

which will tend to zero as $\nu \to \infty$, because

$$f^\nu(z_{ij}) \equiv 0,$$

$$\sum_{j=1}^{m} l_{ij}(s) a(z_{ij}) p_i^{\nu'}(z_{ij}) \to a(s) p_i^{\nu'}(s)$$

and

$$\sum_{j=1}^{m} l_{ij}(s) b(z_{ij}) p_i^\nu(z_{ij}) \to b(s) p_i^\nu(s).$$

As a result, if we integrate the above equation over $[0, s]$ we have

$$\int_0^s \{\mathbf{p}_{\mathbf{h}^\nu}^{''}(x) + a(x)\mathbf{p}_{\mathbf{h}^\nu}^{'}(x) + b(x)\mathbf{p}_{\mathbf{h}^\nu}(x)\}dx \;=$$

$$\mathbf{p}_{\mathbf{h}^\nu}^{'}(s) - \mathbf{p}_{\mathbf{h}^\nu}^{'}(0) + \int_0^s \{a(x)\mathbf{p}_{\mathbf{h}^\nu}^{'}(x) + b(x)\mathbf{p}_{\mathbf{h}^\nu}(x)\}dx.$$

When $\nu \rightarrow \infty$ the above expression goes to zero, and $\mathbf{p_{h^\nu}}(x)$ tends to $p(x)$. Therefore, we can write

$$p^{'}(s) = p^{'}(0) - \int_0^s \{a(x)p^{'}(x) + b(x)p(x)\}dx.$$

From Lemma (3.9.1) we know that $p \in C^1[0,1]$. In addition, since the right hand side of this equation is differentiable, it follows that $p \in C^2[0,1]$. Thus $p^{''}$ exists. By taking the derivative we find

$$p^{''}(s) + a(s)p^{'}(s) + b(s)p(s) = 0.$$

Using the boundary conditions, we have $\mathbf{p_{h^\nu}}(0) = \mathbf{p_{h^\nu}}(1) = 0$, as a result when $\nu \rightarrow \infty$, we have also $p(0) = p(1) = 0$. From the assumption in the statement of the theorem we conclude that $p \equiv 0$, which can be shown to contradict the fact that $\| \mathbf{p_{h^\nu}} \|_1 = 1$.

Proof of the second part: This will also be done by contradiction, similar to the proof of the first part. Suppose, on the contrary, that the second part of the theorem does not hold.

By Part 1, the collocation system, which is a linear system of N*(m+2) unknown coefficients and N*(m+2) equations, for all small $\mid h^\nu \mid$ has a unique solution. As a result, we can find a sequence of meshes $\{h^\nu\}_{\nu=1}^{\infty}$, with $\mid h^\nu \mid \rightarrow 0$ as $\nu \rightarrow \infty$, and for each mesh h^ν quantities f^ν with $\| f^\nu \|_\infty \rightarrow 0$, such that the corresponding unique solution $\mathbf{p_{h^\nu}}$ of the collocation equations

$$Lp_i^\nu(z_{ij}) = f^\nu(z_{ij}), \quad i = 1, \cdots, N, \quad j = 1, \cdots, m, \quad p_1^\nu(0) = p_N^\nu(1) = 0,$$

where, as before, p_i^ν is a part of the piecewise polynomial $\mathbf{p_{h^\nu}}$, satisfies $\| \mathbf{p_{h^\nu}} \|_1 = 1$.

For each mesh h^ν as $\nu \rightarrow \infty$, we have

$$f_{ij}^\nu \equiv f^\nu(z_{ij}) \rightarrow 0 \text{ and } \| f^\nu \|_\infty \rightarrow 0.$$

As $p_i^\nu \in P_{m+2}$ we have $p_i^{\nu''} \in P_m$. Again, $p_i^{\nu''}(x)$ is completely determined by $\{p_i^{\nu''}(z_{ij}) \equiv p_{ij}^{\nu''}\}_{j=1}^m$, and therefore we can write $p_i^{\nu''}$ in terms of Lagrange basis functions for $l_{ij} \in P_m$, $l_{ij}(z_{ij}) = \delta_{ij}$

$$p_i^{\nu''}(x) = \sum_{j=1}^{m} l_{ij}(x)p_{ij}^{\nu''}.$$

The collocation points (z_{ij}) are chosen according to a template. As a result, the $l_{ij}(x)$ are bounded ($l_{ij}(x) \leq K_l$ (a constant)), and we can write

$$\begin{aligned} \max_{x\in[x_{i-1},x_i]} \mid p_i^{\nu''}(x) \mid &= \max \mid \sum_{j=1}^{m} l_{ij}(x) p_{ij}^{\nu''} \mid \\ &\leq mK_l \max_i \mid f^{\nu}(z_{ij}) - a(z_{ij}) p_i^{\nu'}(z_{ij}) - b(z_{ij}) p_i^{\nu}(z_{ij}) \mid . \end{aligned} \tag{3.43}$$

As $f^{\nu}(z_{ij}) \rightarrow 0$, and also $p_i^{\nu'}(z_{ij})$ and $p_i^{\nu}(z_{ij})$ are bounded quantities, we have

$$\max_{x\in[x_{i-1},x_i]} \mid p_i^{\nu''}(x) \mid \leq K \text{ (a constant)}.$$

From Lemma (3.9.1) we conclude that there exist a subsequence of $\{\mathbf{p_h}^{\nu}\}_{\nu=1}^{\infty}$ and a function $p \in C^1[0,1]$ such that:

$$\| \mathbf{p_h}^{\nu} - p \|_1 \rightarrow 0 \quad \text{as} \quad \nu \rightarrow \infty.$$

Take an arbitrary point $s \in [0,1]$ inside an element i, $(s \in [x_{i-1}, x_i])$. We can write

$$\begin{aligned} &p_i^{\nu''}(s) + a(s) p_i^{\nu'}(s) + b(s) p_i^{\nu}(s) = \textstyle\sum_{j=1}^{m} l_{ij}(s) p_{ij}^{\nu''} + a(s) p_i^{\nu'}(s) + b(s) p_i^{\nu}(s) = \\ &\textstyle\sum_{j=1}^{m} l_{ij}(s) \{ f^{\nu}(z_{ij}) - a(z_{ij}) p_i^{\nu'}(z_{ij}) - b(z_{ij}) p_i^{\nu}(z_{ij}) \} + a(s) p_i^{\nu'}(s) + b(s) p_i^{\nu}(s). \end{aligned} \tag{3.44}$$

The above expression will tend to zero as $\nu \rightarrow \infty$, because $\sum_{j=1}^{m} l_{ij}(s) a(z_{ij}) p_i^{\nu'}(z_{ij}) \rightarrow a(s) p_i^{\nu'}(s)$, $\sum_{j=1}^{m} l_{ij}(s) b(z_{ij}) p_i^{\nu}(z_{ij}) \rightarrow b(s) p_i^{\nu}(s)$, and $f^{\nu}(z_{ij}) \rightarrow 0$. If we integrate this equation over $[0, s]$ we have

$$\int_0^s \{ \mathbf{p}_{\mathbf{h}^{\nu}}''(x) + a(x) \mathbf{p}_{\mathbf{h}^{\nu}}'(x) + b(x) \mathbf{p}_{\mathbf{h}^{\nu}}(x) \} dx =$$

$$\mathbf{p}_{\mathbf{h}^{\nu}}'(s) - \mathbf{p}_{\mathbf{h}^{\nu}}'(0) + \int_0^s \{ a(x) \mathbf{p}_{\mathbf{h}^{\nu}}'(x) + b(x) \mathbf{p}_{\mathbf{h}^{\nu}}(x) \} dx.$$

Therefore, when $\nu \rightarrow \infty$ we have

$$p'(s) = p'(0) - \int_0^s \{ a(x) p'(x) + b(x) p(x) \} dx.$$

From Lemma (3.9.1) we have $p \in C^1[0,1]$. Since the right hand side of this equation is differentiable, it follows that $p \in C^2[0,1]$, so that p'' exists. By taking the derivative we have

$$p''(s) + a(s) p'(s) + b(s) p(s) = 0.$$

Using the boundary conditions, we have $p(0) = p(1) = 0$. As in Part 1, from the assumption in the statement of the theorem we conclude that $p \equiv 0$, which can be shown to contradict the fact that $\| \mathbf{p_h}^{\nu} \|_1 = 1$.

3.10 Summary of the complete collocation algorithm

The complete algorithm of the finite element collocation method for linear ODEs is as follows:

- Mesh generation:
  - Recursive subdivision of the domain until the desired mesh size is achieved.
  - Construction of the binary tree to be used in the nested dissection procedure.
- Loop over all elements
  - Compute Φ, L_Φ and R_Φ, using Equations (3.9) and (3.10).
  - Solve Equation (3.11) to find A and B for each element.
- Construct Equation (3.8) for the root node of the binary tree, using the nested dissection procedure described in Section 3.8.
- Use the boundary conditions, together with Equation (3.8) for the root node, to solve for the unknowns on the boundary of the domain.
- Determine the unknowns at the interior mesh points using the recursive back-substitution, as mentioned in Section 3.8.

3.10.1 Stability of the solution algorithm

The stability of the collocation method as established in Section 3.9 for linear model problems, does not take into account the algorithm used for solving the resulting linear equations. In fact, in the analysis we implicitly assume that the linear systems, which are shown to be nonsingular, are set up and solved *exactly*. In practice, the arithmetic is inexact, and moreover the actual algorithm used to set up and solve the linear systems may introduce instabilities. In this section we discuss certain aspects of this fact. The numerical stability of the solution algorithm depends on the stability of the nested dissection procedure. As an example, in Figure 3.5 a simple mesh and its binary tree is shown. The domain is subdivided into two subdomains and each subdomain is subdivided into two elements. The unknowns are

$u_1,\ v_1,\ \cdots,\ u_5,\ v_5$. If we write all collocation equations, together with the matching conditions between adjacent elements without, the insertion of global boundary conditions, we have the following form of linear system:

$$\begin{array}{l} Ia) \\ Ib) \\ IIa) \\ IIb) \\ IIIa) \\ IIIb) \\ IVa) \\ IVb) \end{array} \begin{pmatrix} v_1 \\ v_2 \\ v_2 \\ v_3 \\ v_3 \\ v_4 \\ v_4 \\ v_5 \end{pmatrix} = \begin{pmatrix} x & x & & & & & & \\ x & x & & & & & & \\ & & x & x & & & & \\ & & x & x & & & & \\ & & & & x & x & & \\ & & & & x & x & & \\ & & & & & & x & x \\ & & & & & & x & x \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{pmatrix} + \begin{pmatrix} x \\ x \\ x \\ x \\ x \\ x \\ x \\ x \end{pmatrix}. \tag{3.45}$$

Here the "x"s denote elements that are generally nonzero in the global system. Equations $(I, \cdots, IV)$ represent Equations (3.8) for each element. For solving this system, using the nested dissection method, we eliminate u_2 from Equations (Ia) and (IIb) using Equations (Ib), (IIa) and the matching conditions between elements 3 and 4. We also eliminate u_4 from Equations $(IIIa)$ and (IVb), using Equations $(IIIb)$, (IVa) and the matching conditions between elements 3 and 4. Therefore the system

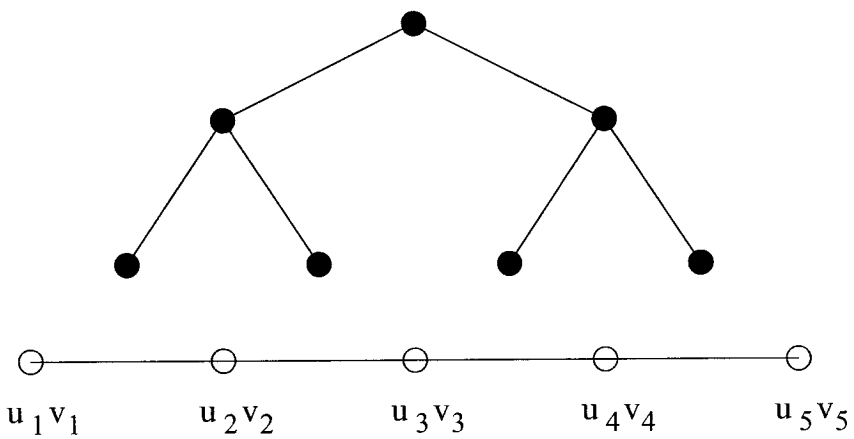


Figure 3.5: A simple mesh with its binary tree

(3.45) changes to the following system:

$$\begin{array}{l} I'a) \\ I'b) \\ II'a) \\ II'b) \\ III'a) \\ III'b) \\ IV'a) \\ IV'b) \end{array} \begin{pmatrix} v_1 \\ v_2 \\ v_2 \\ v_3 \\ v_3 \\ v_4 \\ v_4 \\ v_5 \end{pmatrix} = \begin{pmatrix} x & & y & & \\ x & x & & & \\ & x & x & & \\ y & & x & & \\ & & x & & y \\ & & x & x & \\ & & & x & x \\ & & y & & x \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{pmatrix} + \begin{pmatrix} x \\ x \\ x \\ x \\ x \\ x \\ x \\ x \end{pmatrix}. \tag{3.46}$$

Here the "y"s are new filled elements in the system. Now, eliminate u_3 from Equations $(I'a)$ and $(IV'b)$ using Equations $(II'b)$, $(III'a)$ and the matching conditions between subdomains 1 and 2. Then the system (3.46) changes to the following system:

$$\begin{array}{l} I''a) \\ I''b) \\ II''a) \\ II''b) \\ III''a) \\ III''b) \\ IV''a) \\ IV''b) \end{array} \begin{pmatrix} v_1 \\ v_2 \\ v_2 \\ v_3 \\ v_3 \\ v_4 \\ v_4 \\ v_5 \end{pmatrix} = \begin{pmatrix} x & & & & z \\ x & x & & & \\ & x & x & & \\ y & & x & & \\ & & x & & y \\ & & x & x & \\ & & & x & x \\ z & & & & x \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{pmatrix} + \begin{pmatrix} x \\ x \\ x \\ x \\ x \\ x \\ x \\ x \end{pmatrix}. \tag{3.47}$$

Here the "z"s are new filled elements in the system. The Equations $(I''a)$ and $(IV''b)$ can be solved with global boundary conditions over the whole domain.

As one can see, the stability of this procedure depends on the elimination of common variables in two adjacent regions. We must be able to set up Equations (3.39) and (3.37). Therefore the matrix E must not be singular. Often this matrix is diagonally dominant and invertible. If it is singular, then we can use a Gauss elimination algorithm with a full pivoting strategy, as explained in detail in the next chapter.

Consider the result of the above elimination for two adjacent elements, as shown in Figure 3.6. The global system for these two elements, using Example 3.5, is

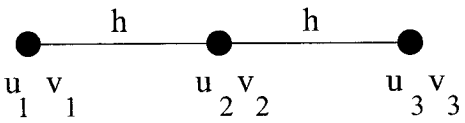


Figure 3.6: Two adjacent 2D elements

$$\begin{pmatrix} v_1 \\ v_2 \\ v_2 \\ v_3 \end{pmatrix} = 1/h \begin{pmatrix} -1 & 1 & & \\ -1 & 1 & & \\ & & -1 & 1 \\ & & -1 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} + h \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix}, \tag{3.48}$$

where $c_1, \cdots, c_4$ are right hand side coefficients. We have

$$E = 1/h - (-1/h) = 2/h.$$

Therefore, we can solve this system using equation (3.39). After elimination of u_3 we have

$$\begin{pmatrix} v_1 \\ v_2 \\ v_2 \\ v_3 \end{pmatrix} = 1/h \begin{pmatrix} -.5 & 0 & .5 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ -.5 & 0 & .5 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} + h \begin{pmatrix} c_1 + .5\ \hat{c} \\ c_2 \\ c_3 \\ c_4 - .5\ \hat{c} \end{pmatrix}, \tag{3.49}$$

where $\hat{c} = c_3 - c_2$.

3.10.2 Example

As another example, consider the following simple second order linear ODE

$$Lu = u''(x) + \lambda u(x) = f(x), \qquad 0 \le x \le 1. \tag{3.50}$$

As boundary conditions take

$$u(0) = u(1) = 0. \tag{3.51}$$

Consider a line segment element as shown in Figure 3.7. The element is centered at x_0 and has a length h. Let $m = 1$ and take the collocation point z to be at the center of the element, *i.e.*, $z = x_0$. Corresponding to the matching points $(x_0 - h/2, x_0 + h/2)$ the approximation in Equations (3.5) takes the form

$$\begin{aligned} v_{x_0-h/2} &= \alpha_{11} u_{x_0-h/2} + \alpha_{12} u_{x_0+h/2} + \beta_1 f(z_1), \\ v_{x_0+h/2} &= \alpha_{21} u_{x_0-h/2} + \alpha_{22} u_{x_0+h/2} + \beta_2 f(z_1). \end{aligned} \tag{3.52}$$

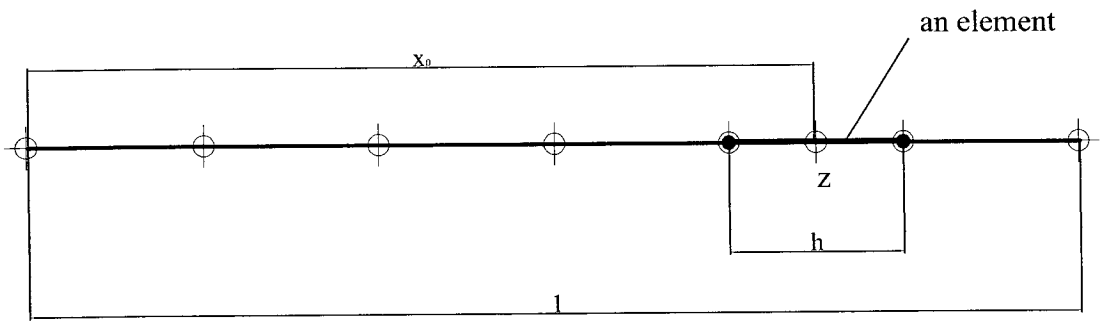


Figure 3.7: A 2D element with center at x_0

As basis of $P_{2+m} = P_3$ choose polynomials $\{1, x, x^2\}$. The coefficients α_{ij} and β_i are obtained by solving Equation (3.11), which is

$$\begin{pmatrix} 1 & 1 & \lambda \\ x_0 - h/2 & x_0 + h/2 & \lambda x_0 \\ (x_0 - h/2)^2 & (x_0 + h/2)^2 & 2 + \lambda x_0^2 \end{pmatrix} \begin{pmatrix} \alpha_{11} & \alpha_{21} \\ \alpha_{12} & \alpha_{22} \\ \beta_1 & \beta_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 2x_0 - h & 2x_0 + h \end{pmatrix}.$$

Upon solving this system, we find:

$$A = \frac{1}{\lambda h^3 - 8h} \begin{pmatrix} 8 - 3\lambda h^2 & -8 - \lambda h^2 \\ 8 + 3\lambda h^2 & -8 + 3\lambda h^2 \end{pmatrix}, \qquad B = \frac{4h}{\lambda h^2 - 8} \begin{pmatrix} 1 \\ -1 \end{pmatrix}. \tag{3.53}$$

Take $f(x) \equiv 0$. For two adjacent elements in the nested dissection procedure we have:

$$\begin{pmatrix} v_1 \\ v_2 \\ v_2 \\ v_3 \end{pmatrix} = \frac{1}{\lambda h^3 - 8h} \begin{pmatrix} 8 - 3\lambda h^2 & -8 - \lambda h^2 & & \\ 8 + 3\lambda h^2 & -8 + 3\lambda h^2 & & \\ & & 8 - 3\lambda h^2 & -8 - \lambda h^2 \\ & & 8 + 3\lambda h^2 & -8 + 3\lambda h^2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}. \tag{3.54}$$

We have

$$E = \frac{-8 + 3\lambda h^2 - (8 - 3\lambda h^2)}{\lambda h^3 - 8h} = \frac{-16 + 6\lambda h^2}{\lambda h^3 - 8h}.$$

Now note that if $\lambda = 8/(3h^2)$ then $E = 0$, and we cannot solve this system using the nested dissection method as described in Section 3.8. In this case the system

becomes:

$$\begin{pmatrix} v_1 \\ v_2 \\ v_2 \\ v_3 \end{pmatrix} = \frac{3}{16h} \begin{pmatrix} 0 & -32/3 & & \\ 32/3 & 0 & & \\ & 0 & -32/3 & \\ & 32/3 & 0 & \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}.$$

Nevertheless, with proper boundary condition, this system is solvable. To avoid cases like this, we will use a modified nested dissection method with a pivoting strategy, that we will present in the next chapter.

3.11 Special basis functions

3.11.1 Selection of the basis functions

For each finite element, the system (3.11) must be solved for the coefficients α_{ij} and β_{ij} . If the basis functions are chosen properly, then this procedure can be done efficiently. Special basis functions help us to change the above system into a form that can be solved easily. In Section 3.6 we have used Lagrange basis functions, and we divided the basis of P_{m+2} into two groups, namely, $\{\phi_1, \phi_2\}$ and $\{\psi_1, \cdots, \psi_m\}$. As a result Equation (3.11) changes to Equation (3.17) or

$$\begin{pmatrix} \Phi & L_\Phi \\ \Psi & L_\Psi \end{pmatrix} \begin{pmatrix} A^T \\ B^T \end{pmatrix} = \begin{pmatrix} R_\Phi \\ R_\Psi \end{pmatrix}, \tag{3.55}$$

where now the $\tilde{}$ has be omitted for simplicity of notation. Evaluating Φ and Ψ, we have

$$\begin{pmatrix} I & L_\Phi \\ O & L_\Psi \end{pmatrix} \begin{pmatrix} A^T \\ B^T \end{pmatrix} = \begin{pmatrix} R_\Phi \\ R_\Psi \end{pmatrix},$$

which can be solved efficiently using Equations (3.19) and (3.20).

In many particular cases one can construct basis functions such that

$$\begin{pmatrix} \Phi & \Phi'' \\ \Psi & \Psi'' \end{pmatrix} = \begin{pmatrix} I & O \\ O & I \end{pmatrix}. \tag{3.56}$$

This is a special case of the system (3.55), namely, when $L = d^2/dx^2$, in which case the system has the explicit solution $A = R_\Phi^T$ and $B = R_\Psi^T$. If we construct special basis functions that satisfy Equation (3.56), then we can easily evaluate coefficients of L_Φ and L_Ψ in the general case because $\phi_i''(x_j) = 0$ and $\psi_i''(x_j) = \delta_{ij}$. In this case

$$[L_\Phi]_{ij} = L\phi_i(z_j) \quad = \quad \phi_i''(z_j) + a(z_j)\phi_i'(z_j) + b(z_j)\phi_i(z_j)$$

$$= a(z_j)\phi_i'(z_j) + b(z_j)\phi_i(z_j), \tag{3.57}$$

$$\begin{aligned}[L_\Psi]_{ij} = L\psi_i(z_j) &= \psi_i''(z_j) + a(z_j)\psi_i'(z_j) + b(z_j)\psi_i(z_j) \\ &= \delta_{ij} + a(z_j)\psi_i'(z_j) + b(z_j)\psi_i(z_j). \end{aligned} \tag{3.58}$$

where δ_{ij} denotes the Kronecker delta function.

3.11.2 Evaluation of the weights

Using the special basis functions satisfying Equation (3.56), the coefficients α_{ij} and β_{ij}, *i.e.*, the matrices A and B, can be obtained from Equations (3.19) and (3.20). This requires evaluation of L_Φ, L_Ψ, R_Φ and R_Ψ, *i.e.*, we need the values of ϕ_i, ϕ_i', ψ_i, and ψ_i' at the collocation points z_j, and the values of ϕ_i' and ψ_i' at the matching points x_j. These quantities will be called *weights*. Note that second derivatives are not needed. If the discretization is uniform then each finite element has the same set of weights. Even if the discretization is not uniform, then the weights of finite elements are related by a scaling factor (when the collocation points are chosen using a template). Thus, in these cases, the weights can be pre-computed.

We now show in detail how to pre-compute the weights $\phi_i(z_j)$ and $\psi_i(z_j)$. Let θ_i, $i = 1, \cdots, m+2$, denote the monomial basis of P_{m+2}. For example, we can take $P_3 = Span\{1, x, x^2\}$. Monomials and their derivatives can be evaluated efficiently. We can write

$$\phi_i(x) = \sum_{k=1}^{m+2} c_{ik}\theta_k(x) \quad \text{and} \quad \psi_i(x) = \sum_{k=1}^{m+2} d_{ik}\theta_k(x),$$

for certain constants c_{ij} and d_{ij}. Using Equation (3.56) we have

$$\begin{aligned} \phi_i(x_j) &= \delta_{ij}, \quad \phi_i''(z_j) = 0, \\ \psi_i(x_j) &= 0, \quad \psi_i''(z_j) = \delta_{ij}, \end{aligned}$$

that is,

$$\begin{aligned} \textstyle\sum_{k=1}^{m+2} c_{ik}\theta_k(x_j) &= \delta_{ij}, \quad \sum_{k=1}^{m+2} c_{ik}\theta_k''(z_j) = 0, \\ \textstyle\sum_{k=1}^{m+2} d_{ik}\theta_k(x_j) &= 0, \quad \sum_{k=1}^{m+2} d_{ik}\theta_k''(z_j) = \delta_{ij}. \end{aligned} \tag{3.59}$$

We define

$$\Theta = \begin{pmatrix} \theta_1(x_{i-1}) & \theta_1(x_i) \\ \vdots & \vdots \\ \theta_{m+2}(x_{i-1}) & \theta_{m+2}(x_i) \end{pmatrix}, \qquad \Delta_\Theta = \begin{pmatrix} \theta_1''(z_1) & \cdots & \theta_1''(z_m) \\ \vdots & & \vdots \\ \theta_{m+2}''(z_1) & \cdots & \theta_{m+2}''(z_m) \end{pmatrix},$$

and

$$C = \begin{pmatrix} c_{11} & \cdots & c_{1,m+2} \\ c_{21} & \cdots & c_{2,m+2} \end{pmatrix}, \qquad D = \begin{pmatrix} d_{11} & \cdots & d_{1,m+2} \\ \vdots & & \vdots \\ d_{m1} & \cdots & d_{m,m+2} \end{pmatrix}.$$

Then Equation(3.59) can be written as

$$\begin{pmatrix} \Theta^T \\ \Delta_\Theta^T \end{pmatrix} (C^T \mid D^T) = \begin{pmatrix} I & O \\ O & I \end{pmatrix}.$$

Thus

$$(C^T \mid D^T) = \begin{pmatrix} \Theta^T \\ \Delta_\Theta^T \end{pmatrix}^{-1}. \tag{3.60}$$

To evaluate $\phi_i(z_j)$ and $\psi_i(z_j)$ we have

$$\phi_i(z_j) = \sum_{k=1}^{m+2} c_{ik}\theta_k(z_j), \quad \text{and} \quad \psi_i(z_j) = \sum_{k=1}^{m+2} d_{ik}\theta_k(z_j). \tag{3.61}$$

Defining

$$\Phi_z = \begin{pmatrix} \phi_1(z_1) & \cdots & \phi_1(z_m) \\ \phi_2(z_1) & \cdots & \phi_2(z_m) \end{pmatrix}, \qquad \Psi_z = \begin{pmatrix} \psi_1(z_1) & \cdots & \psi_1(z_m) \\ \vdots & & \vdots \\ \psi_m(z_1) & \cdots & \psi_m(z_m) \end{pmatrix},$$

and

$$\Theta_z = \begin{pmatrix} \theta_1(z_1) & \cdots & \theta_1(z_m) \\ \vdots & & \vdots \\ \theta_{m+2}(z_1) & \cdots & \theta_{m+2}(z_m) \end{pmatrix},$$

we can write Equation (3.61) as

$$\begin{pmatrix} \Phi_z \\ \Psi_z \end{pmatrix} = \begin{pmatrix} C \\ D \end{pmatrix} \Theta_z,$$

or

$$(\Phi_z^T \mid \Psi_z^T) = \Theta_z^T (C^T \mid D^T).$$

Now, using Equation (3.60), it follows that

$$(\Phi_z^T \mid \Psi_z^T) = \Theta_z^T \begin{pmatrix} \Theta^T \\ \Delta_\Theta^T \end{pmatrix}^{-1},$$

from which

$$(\Phi_z^T \mid \Psi_z^T) \begin{pmatrix} \Theta^T \\ \Delta_\Theta^T \end{pmatrix} = \Theta_z^T,$$

or

$$(\Theta \mid \Delta_\Theta) \begin{pmatrix} \Phi_z \\ \Psi_z \end{pmatrix} = \Theta_z. \tag{3.62}$$

Thus the weights $\phi_i(z_j)$ and $\psi_i(z_j)$ can be computed by solving the $n+m$ by $n+m$ System (3.62) for Φ_z and Ψ_z.

Other weights, namely, ϕ_i' and ψ_i', evaluated at the collocation points z_j and at the matching points x_j, can be computed similarly, using the same $m+2$ by $m+2$ matrix. Thus only one LU-decomposition is needed. Furthermore, as mentioned earlier, weights for similar finite elements can be obtained by scaling.

Chapter 4

The Collocation Method for Nonlinear ODE BVP

4.1 Introduction

In preparation for the case of nonlinear PDEs, we consider in this section the finite element collocation method for nonlinear BVP ODEs. The finite difference approach, as described in Section 3.4, cannot be used for nonlinear problems, because there is no linear relation of the form Equation (3.5), or, equivalently, Equation (3.8), between the variable u and its derivative v at the boundary points (matching points) of an element. Instead, the collocation formulation, described in Section 3.7 for the linear case, is the appropriate approach for the nonlinear case. Nevertheless, the nested dissection procedure, which in the linear case was suggested by the finite difference formulation, can be generalized to the nonlinear case, as will be explained below, namely, in the solution of the linearized Newton systems.

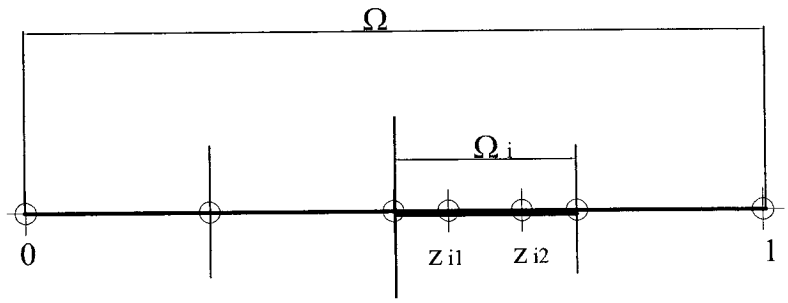


Figure 4.1: Recursive subdivision and a finite element mesh

4.2 Collocation with continuous piecewise polynomials

Consider the second order nonlinear ODE

$$Nu = u''(x) + f(x, u, u') = 0, \qquad 0 \leq x \leq 1, \tag{4.1}$$

where $f(.), u(x) \in \mathcal{R}$. As boundary conditions, take

$$u(0) = u(1) = 0. \tag{4.2}$$

As in the previous chapter, this example uses simple boundary conditions, but it is possible to treat more general boundary conditions. For simplicity and ease of presentation of the method, we choose the line segment $0 \leq x \leq 1$ in Figure 4.1 as the domain Ω of the problem. We construct a mesh using recursive subdivision of the domain, as already presented in the preceding chapter. The domain, which is the root region, is recursively subdivided into two sub-regions. This subdivision procedure is continued until a desired level of mesh has been achieved. The smallest regions are the finite elements, which correspond to leaf nodes in the binary recursion tree.

4.3 The finite element collocation formulation

Consider any finite element Ω_i, as illustrated in Figure 4.1. We choose collocation points z_{ik}, $k = 1, \cdots, m$ inside this element. A polynomial $p(x) \in P_{m+2}$, associated to each finite element, is required to satisfy the collocation equations

$$Np(z_k) = 0, \qquad k = 1, \cdots, m.$$

(For notational simplicity, the index i from the above equation is dropped.) For any two adjacent elements, the values and the derivatives of the neighboring polynomials are required to match at their boundary (matching) point.

For each finite element the local polynomial has the form

$$p(x) = \sum_{i=1}^{m+2} c_i \phi_i(x)$$

where $Span\{\phi_1, \cdots, \phi_{m+2}\} = P_{m+2}$. The collocation equations are

$$N(\sum_{i=1}^{m+2} c_i \phi_i(z_k)) = 0, \qquad k = 1, \cdots, m. \tag{4.3}$$

To impose the continuity requirements, unique variables u_i and v_i are associated to each matching point x_i on the common boundary points, namely,

$$\begin{aligned} p(x_i) &= u_i, \\ p(x_i)' &= v_i. \end{aligned}$$

Using the notation of the previous chapter, we can write

$$\begin{aligned} u - \Phi^T c &= 0, \\ v - R_\Phi^T c &= 0. \end{aligned} \tag{4.4}$$

where

$$u = (u_{i-1}, u_i)^T, \qquad v = (v_{i-1}, v_i)^T,$$

and

$$\Phi = \begin{pmatrix} \phi_1(x_{i-1}) & \phi_1(x_i) \\ \vdots & \vdots \\ \phi_{m+2}(x_{i-1}) & \phi_{m+2}(x_i) \end{pmatrix}, \quad R_\Phi = \begin{pmatrix} \phi_1'(x_{i-1}) & \phi_1'(x_i) \\ \vdots & \vdots \\ \phi_{m+2}'(x_{i-1}) & \phi_{m+2}'(x_i) \end{pmatrix}. \tag{4.5}$$

4.4 Newton's method

Equations (4.3) and (4.4), together with the discrete boundary conditions, constitute the discretization. The unknowns are $c \in R^{m+2}$, for each finite element; and the u_i and v_i associated with the points x_i on inter-element boundaries (the end points of the elements). To solve Equations (4.3) and (4.4) for u, v, and c, we use Newton's method. Omitting iteration indices, it can be written as

$$L_\Phi^T \, \delta c = -r_N, \tag{4.6}$$

$$\begin{aligned} &\text{a)} \quad \delta u - \Phi^T \delta c = -r_u, \\ &\text{b)} \quad \delta v - R_\Phi^T \delta c = -r_v. \end{aligned} \tag{4.7}$$

with

$$L_\Phi^T = \begin{pmatrix} L[p(z_1)]\phi_1(z_1) & \cdots & L[p(z_1)]\phi_{m+2}(z_1) \\ \vdots & & \vdots \\ L[p(z_m)]\phi_1(z_m) & \cdots & L[p(z_m)]\phi_{m+2}(z_m) \end{pmatrix}, \tag{4.8}$$

where L is the linearization of N, *i.e.*, $L[p]\phi(z)$ is the linearization of N about p acting on ϕ at z. More precisely, we can write

$$L[p]\phi(z) = \phi''(z) + D_2 f(z, p(z), p'(z))\phi(z) + D_3 f(z, p(z), p'(z))^T \phi'(z).$$

Here, $D_2 f(x, u, u')$ is the derivative of f with respect to u and $D_3 f(x, u, u')$ is the derivative of f with respect to u'. Further we define

$$\delta c = \begin{pmatrix} \delta c_1 \\ \vdots \\ \delta c_{m+2} \end{pmatrix}, \quad r_N = \begin{pmatrix} Np(z_1) \\ \vdots \\ Np(z_m) \end{pmatrix}, \quad \delta u = \begin{pmatrix} \delta u_{i-1} \\ \delta u_i \end{pmatrix}, \quad \delta v = \begin{pmatrix} \delta v_{i-1} \\ \delta v_i \end{pmatrix}, \tag{4.9}$$

and

$$r_u = u - \Phi^T c, \qquad r_v = v - R_\Phi^T c.$$

Equations (4.6) and (4.7a) can be written

$$\begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix} \delta c = \begin{pmatrix} \delta u + r_u \\ -r_N \end{pmatrix}. \tag{4.10}$$

Using Equation (4.10) to eliminate δc in Equation (4.7b) one obtains

$$\delta v = R_\Phi^T \begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix}^{-1} \begin{pmatrix} \delta u + r_u \\ -r_N \end{pmatrix} - r_v.$$

Define A and B precisely as in the previous chapter,

$$A = \begin{pmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{12} & \alpha_{22} \end{pmatrix}, \qquad B = \begin{pmatrix} \beta_{11} & \cdots & \beta_{1m} \\ \beta_{21} & \cdots & \beta_{2m} \end{pmatrix},$$

which are the solutions of the following linear system

$$(\Phi \mid L_\Phi) \begin{pmatrix} A^T \\ B^T \end{pmatrix} = R_\Phi. \tag{4.11}$$

The above expression for δv can be rewritten as

$$\delta v = (A \quad B) \begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix} \begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix}^{-1} \begin{pmatrix} \delta u + r_u \\ -r_N \end{pmatrix} - r_v,$$

that is,

$$\delta v = A\delta u - Br_N - r_v + Ar_u. \tag{4.12}$$

It is important to note that this equation has the same form as Equation (3.8) of the previous chapter. As a result, the nested dissection procedure used for linear ODEs can also be used here. As mentioned in Section 3.10.1, the original nested dissection procedure, as described in Section 3.8, can fail due to possible singularity of the matrix E in Equation (3.37). In the next section, we introduce a variation on the nested dissection method.

4.4.1 Modified nested dissection

We use a nested dissection procedure to eliminate the unknowns δu and δv on boundaries separating adjacent regions. The procedure results in the elimination of all interior unknowns. One is left with one equation for each x_i on $\delta\Omega$ (the end points of the domain). Subsequently the boundary conditions can be used to determine the values of δu and δv at the boundaries. Thereafter, a recursive back-substitution gives the values of δu (and hence δv) at each interior point common to adjacent regions.

Consider two adjacent regions Ω_1 and Ω_2 as in Figure 3.4. The elimination of unknowns δu and δv on the common boundary is done as in the domain decomposition. The common boundary is called $\delta\Omega_{12}$, and the remaining parts of the boundaries of Ω_1 and Ω_2 are called $\delta\Omega_1$ and $\delta\Omega_2$ respectively. The vector δu in the collocation formulation (4.12) is split accordingly into δu_1 and δu_{12} for region Ω_1 and into δu_{21} $(= \delta u_{12})$ and δu_2 for region Ω_2. Vector δv , r_u, r_v and r_N are similarly split.

To prevent the problem mentioned in Section 3.10.1 (the nested dissection procedure can fail due to singularity of the matrix E in Equation (3.37)), we present a modified procedure. First, to have a more flexible system, we rewrite Equation (4.12) in the form

$$D\delta v = A\delta u + g, \tag{4.13}$$

where

$$g = -Br_N - r_v + Ar_u.$$

Matrix D is the identity matrix at the element level (at leaves of the binary tree), but it is not necessary the identity matrix at the interior nodes of the binary tree, during

the execution of the method. Equation (4.13) can be written in the split form as

$$\begin{pmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{pmatrix} \begin{pmatrix} \delta v_1 \\ \delta v_{12} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} \delta u_1 \\ \delta u_{12} \end{pmatrix} + \begin{pmatrix} g_1 \\ g_{12} \end{pmatrix}. \tag{4.14}$$

Matrices D and A have been split into four parts, according to the dimensions of u_1 and u_{12}. Equation (4.12) for region Ω_1 can be written as

$$\begin{pmatrix} D^1_{11} & D^1_{12} \\ D^1_{21} & D^1_{22} \end{pmatrix} \begin{pmatrix} \delta v_1 \\ \delta v_{12} \end{pmatrix} = \begin{pmatrix} A^1_{11} & A^1_{12} \\ A^1_{21} & A^1_{22} \end{pmatrix} \begin{pmatrix} \delta u_1 \\ \delta u_{12} \end{pmatrix} + \begin{pmatrix} g^1_1 \\ g^1_{12} \end{pmatrix}, \tag{4.15}$$

where

$$g^1_1 = -B^1_1 r^1_N - r_{v1} + A^1_{11} r_{u1} + A^1_{12} r_{u12},$$

$$g^1_{12} = -B^1_2 r^1_N - r_{v12} + A^1_{21} r_{u1} + A^1_{22} r_{u12}.$$

For region Ω_2, we have

$$\begin{pmatrix} D^2_{11} & D^2_{12} \\ D^2_{21} & D^2_{22} \end{pmatrix} \begin{pmatrix} \delta v_2 \\ \delta v_{21} \end{pmatrix} = \begin{pmatrix} A^2_{11} & A^2_{12} \\ A^2_{21} & A^2_{22} \end{pmatrix} \begin{pmatrix} \delta u_2 \\ \delta u_{21} \end{pmatrix} + \begin{pmatrix} g^2_2 \\ g^2_{21} \end{pmatrix}, \tag{4.16}$$

where

$$g^2_2 = -B^2_1 r^2_N - r_{v2} + A^2_{11} r_{u2} + A^2_{12} r_{u21},$$

$$g^2_{21} = -B^2_2 r^2_N - r_{v21} + A^2_{21} r_{u2} + A^2_{22} r_{u21}.$$

The superscript represents the region number. As the values and the derivatives of the neighboring polynomials must match (continuity relations), we have

$$\delta u_{12} = \delta u_{21}, \tag{4.17}$$

$$\delta v_{12} = \delta v_{21}.$$

As a result, Equations (4.15), and (4.16) together can be written as follows:

$$\begin{pmatrix} D^1_{11} & 0 & D^1_{12} \\ 0 & D^2_{11} & D^2_{12} \\ D^1_{21} & 0 & D^1_{22} \\ 0 & D^2_{21} & D^2_{22} \end{pmatrix} \begin{pmatrix} \delta v_1 \\ \delta v_2 \\ \delta v_{12} \end{pmatrix} = \begin{pmatrix} A^1_{11} & 0 & A^1_{12} \\ 0 & A^2_{11} & A^2_{12} \\ A^1_{21} & 0 & A^1_{22} \\ 0 & A^2_{21} & A^2_{22} \end{pmatrix} \begin{pmatrix} \delta u_1 \\ \delta u_2 \\ \delta u_{12} \end{pmatrix} + \begin{pmatrix} g^1_1 \\ g^2_2 \\ g^1_{12} \\ g^2_{21} \end{pmatrix}, \tag{4.18}$$

or

$$\begin{pmatrix} D_{22}^1 & -A_{22}^1 & D_{21}^1 & -A_{21}^1 & 0 & 0 \\ D_{22}^2 & -A_{22}^2 & 0 & 0 & D_{21}^2 & -A_{21}^2 \\ D_{12}^1 & -A_{12}^1 & D_{11}^1 & -A_{11}^1 & 0 & 0 \\ D_{12}^2 & -A_{12}^2 & 0 & 0 & D_{11}^2 & -A_{11}^2 \end{pmatrix} \begin{pmatrix} \delta v_{12} \\ \delta u_{12} \\ \delta v_1 \\ \delta u_1 \\ \delta v_2 \\ \delta u_2 \end{pmatrix} = \begin{pmatrix} g_{12}^1 \\ g_{21}^2 \\ g_1^1 \\ g_2^2 \end{pmatrix}. \tag{4.19}$$

Now, we use a row and column pivoting strategy over the first and the second columns of System (4.19) to eliminate all coefficients in rows three and four, and columns one and two. We also transform the square matrices D_{22}^1 and $-A_{22}^2$ to identity matrices (I), and eliminate all coefficients in D_{22}^2 and A_{22}^1. Note that, in our scalar ODE case, D_{22}^1, A_{22}^2, D_{22}^2 and A_{22}^1 are scalars and we must transform D_{22}^1 and $-A_{22}^2$ to 1 and D_{22}^2 and A_{22}^1 to 0. (As we shall see in later chapters, in case of PDEs, D_{22}^1, A_{22}^2, D_{22}^2 and A_{22}^1 are square matrices.) This elimination changes System (4.19) into the following form:

$$\begin{pmatrix} I & 0 & \hat{D}_{11} & \hat{A}_{11} & \hat{D}_{12} & \hat{A}_{12} \\ 0 & I & \hat{D}_{21} & \hat{A}_{21} & \hat{D}_{22} & \hat{A}_{22} \\ 0 & 0 & D_{11} & C_{11} & D_{12} & C_{12} \\ 0 & 0 & D_{21} & C_{21} & D_{22} & C_{22} \end{pmatrix} \begin{pmatrix} \delta v_{12} \\ \delta u_{12} \\ \delta v_1 \\ \delta u_1 \\ \delta v_2 \\ \delta u_2 \end{pmatrix} = \begin{pmatrix} \hat{g}_{12} \\ \hat{g}_{21} \\ \hat{g}_1 \\ \hat{g}_2 \end{pmatrix}. \tag{4.20}$$

From Equation (4.20),we can write

$$\delta u_{12} = \hat{g}_{21} - (\hat{D}_{21}\delta v_1 + \hat{A}_{21}\delta u_1 + \hat{D}_{22}\delta v_2 + \hat{A}_{22}\delta u_2), \tag{4.21}$$

$$\delta v_{12} = \hat{g}_{12} - (\hat{D}_{11}\delta v_1 + \hat{A}_{11}\delta u_1 + \hat{D}_{12}\delta v_2 + \hat{A}_{12}\delta u_2), \tag{4.22}$$

and

$$\begin{pmatrix} D_{11} & C_{11} & D_{12} & C_{12} \\ D_{21} & C_{21} & D_{22} & C_{22} \end{pmatrix} \begin{pmatrix} \delta v_1 \\ \delta u_1 \\ \delta v_2 \\ \delta u_2 \end{pmatrix} = \begin{pmatrix} \hat{g}_1 \\ \hat{g}_2 \end{pmatrix}. \tag{4.23}$$

Equation (4.23) represents the discrete equations for the enlarged region, that is, for the union of Ω_1 and Ω_2, after the elimination of common boundary unknowns δu_{12} and δv_{12}. Note that these new equations are again of the form Equation (4.13).

4.4.2 Complexity

Since for the case of scalar ODEs the coefficients of System (4.19) are scalars, we need approximately the same number of operations in each nested dissection step. Differences in the number of operations are due to the pivoting strategy to transform System (4.19) to System (4.20), *i.e.*, the number of operations depends on the values of coefficients stored in the system. We can consider the maximum possible number of operations, or an average number of operations, needed in each step. These numbers are constant and they do not depend in the position of the node (region) in the binary tree. They depend on the dimension of System (4.19), which, for ODEs, is the same in each step of the nested dissection. Therefore, the complexity of the nested dissection algorithm for ODEs depends only on the number of interior nodes of the binary tree. For a mesh of N elements, we must construct a binary tree with $N-1$ interior nodes. As a result, the complexity of the nested dissection algorithm for an ODE problem is $\mathcal{O}(N)$.

4.5 Special basis functions

For each finite element, System (4.11) must be solved for the coefficients α_{ij} and β_{ij} . This procedure can be done efficiently if we use special basis functions. If the discretization is uniform or if there are many similar finite elements then the Lagrange basis functions defined in Section 3.6 can also be used in the nonlinear case to reduce the number of operations. Using the notation of the previous chapter we can write

$$p(x) = \sum_{i=1}^{2} c_i \phi_i(x) + \sum_{i=1}^{m} d_i \psi_i(x).$$

We divide the basis of P_{m+2} into two groups, namely $\{\phi_1, \phi_2\}$ and $\{\psi_1, \cdots, \psi_m\}$. If we rewrite the matrices Φ, L_Φ and R_Φ defined in Equations (4.5) and (4.8) as presented in Sections 3.6, and 3.11.1, then Equation (4.10) can be written as

$$\begin{pmatrix} \Phi^T & \Psi^T \\ L_\Phi^T & L_\Psi^T \end{pmatrix} \begin{pmatrix} \delta c \\ \delta d \end{pmatrix} = \begin{pmatrix} \delta u + r_u \\ -r_N \end{pmatrix}, \tag{4.24}$$

where $\Phi = I$ and $\Psi = O$. Thus

$$\begin{aligned} \delta c &= \delta u + r_u, \\ L_\Psi^T \delta d &= -r_N - L_\Phi^T \delta c. \end{aligned} \tag{4.25}$$

Equation (4.11) becomes

$$\begin{pmatrix} I & L_\Phi \\ O & L_\Psi \end{pmatrix} \begin{pmatrix} A^T \\ B^T \end{pmatrix} = \begin{pmatrix} R_\Phi \\ R_\Psi \end{pmatrix}.$$

From this equation, we have,

$$\begin{aligned} L_\Psi B^T &= R_\Psi, \\ A^T &= R_\Phi - L_\Phi B^T. \end{aligned} \tag{4.26}$$

As a result the form of Equation (4.12) remains unchanged:

$$\delta v = A\delta u - Br_N - r_v + Ar_u, \tag{4.27}$$

but now we have

$$r_u = u - \begin{pmatrix} \Phi \\ \Psi \end{pmatrix}^T \begin{pmatrix} c \\ d \end{pmatrix} = u - c, \tag{4.28}$$

$$r_v = v - \begin{pmatrix} R_\Phi \\ R_\Psi \end{pmatrix}^T \begin{pmatrix} c \\ d \end{pmatrix} = v - R_\Phi^T c - R_\Psi^T d. \tag{4.29}$$

4.6 Summary of the collocation algorithm

The algorithm of the finite element collocation method for nonlinear ODEs can be summarized as follows:

- Mesh generation as presented in Section 3.10.
- Given current approximations to u, v, and c
- Do Newton's iterations until the error criteria are satisfied.
  - Loop over all elements
    - Compute the matrices Φ, L_Φ and R_Φ using Equations (4.5) and (4.8).
    - Solve Equation (4.11) to find A and B for each element.
    - Compute r_u, r_v and r_N, and write Equations (4.13) for each element.
  - Solve the global system.
    - Construct System (4.13) for the root node of the binary tree, using the nested dissection procedure described in Section 4.4.1.

* Use the Newton equations corresponding to boundary conditions together with the above system to solve for the unknowns, δu and δv on the boundary of the domain.
* Find the unknowns at the interior matching points of the binary tree, using the recursive back-substitution described in Section 4.4.1.

- For each finite element compute δc using Equation (4.10).
- Update : $u \rightarrow u + \delta u$, $v \rightarrow v + \delta v$, and $c \rightarrow c + \delta c$.
- Find the relative errors in the calculation of u, v and c.

- Save the results.

Note that for any given finite element the matrix on the left hand side of (4.10) is the transpose of the matrix on the left hand side of (4.11). Thus only one LU-decomposition is required per finite element.

If Lagrange basis functions are used, the above algorithm is modified as follows:

- Mesh generation as presented in Section 3.10.
- Given current approximations to u, v, c and d.
- Do Newton's iterations until the error criteria are satisfied.
  - Loop over all elements
    * Compute the matrices L_Φ, L_Ψ, R_Ψ and R_Φ, using Equations (4.5) and (4.8).
    * Solve Equation (4.26) to find A and B for each element.
    * Compute r_u, r_v, and r_N, using Equations (4.28), (4.29) and (4.9), and write Equations (4.13) for each element.
  - Solve the global system for the unknowns, δu and δv, as presented in the previous algorithm in this section.
  - For each finite element compute δc and δd using Equation (4.25).
  - Update : $u \rightarrow u + \delta u$, $v \rightarrow v + \delta v$, $c \rightarrow c + \delta c$, and $d \rightarrow d + \delta d$.
  - Compute the relative errors in the calculation of u, v, c and d.

- Save the results.

Again, in Equations (4.26) and (4.25) only one LU-decomposition of the matrix L_Ψ is needed per finite element.

4.6.1 Stability of the solution algorithm

As in the previous chapter, the stability of the solution depends on the stability of the solution algorithm for solving the linearized Newton equation globally. Here, we consider how the system of equations is solved in each Newton's iteration, using the modified nested dissection method presented in Section 4.4.1. For clarity, the simple mesh and its binary tree as presented in Figure 4.2 are considered. The domain is subdivided into two sub-domains and each sub-domain is subdivided into two elements. The overall unknowns are δu_1, δv_1, $\cdots$, δu_5, δv_5. If we write all the collocation equations together, and apply the matching conditions between adjacent elements, without yet taking into account the global boundary conditions, then we have the schematic representation of the global linear system shown in Equation (4.30)

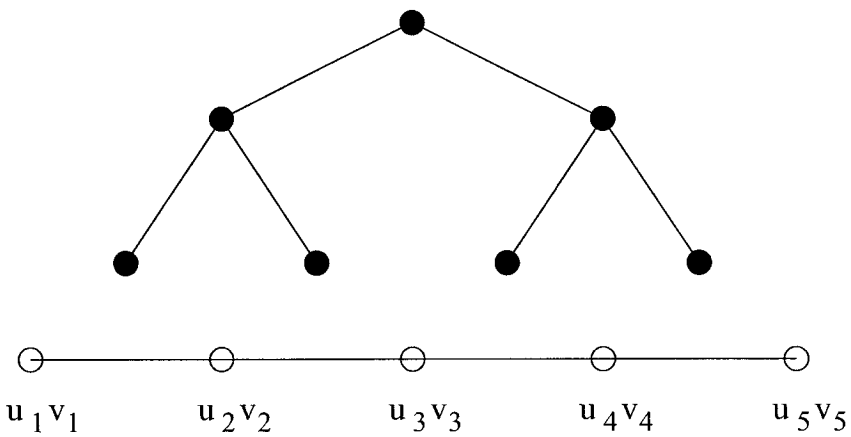


Figure 4.2: A simple mesh with its binary tree

$$
\begin{array}{c}
Ia) \\ Ib) \\ IIa) \\ IIb) \\ IIIa) \\ IIIb) \\ IVa) \\ IVc)
\end{array}
\begin{pmatrix}
x & x & x & x & & & & & & \\
x & x & x & x & & & & & & \\
 & & x & x & x & x & & & & \\
 & & x & x & x & x & & & & \\
 & & & & x & x & x & x & & \\
 & & & & x & x & x & x & & \\
 & & & & & & x & x & x & x \\
 & & & & & & x & x & x & x
\end{pmatrix}
\begin{pmatrix}
\delta u_1 \\ \delta v_1 \\ \delta u_2 \\ \delta v_2 \\ \delta u_3 \\ \delta v_3 \\ \delta u_4 \\ \delta v_4 \\ \delta u_5 \\ \delta v_5
\end{pmatrix}
=
\begin{pmatrix}
x \\ x \\ x \\ x \\ x \\ x \\ x \\ x
\end{pmatrix}. \qquad (4.30)
$$

Here the "x"s are generally nonzero elements in the global system. Equations (I, $\cdots$, IV) represent Equation (4.13), or its equivalent (4.23), for the elements 1, 2, 3, and 4, respectively. Using the modified nested dissection algorithm, we first eliminate δu_2 and δv_2 from Equations (II), using Equations (I) and full pivoting over columns 3 and 4. Then we eliminate δu_4 and δv_4 from Equations (IV), using Equations (III) and full pivoting over columns 7 and 8. As a result the system (4.30) changes into the following form:

$$
\begin{array}{c}
I'a) \\ I'b) \\ II'a) \\ II'b) \\ III'a) \\ III'b) \\ IV'a) \\ IV'b)
\end{array}
\begin{pmatrix}
x & x & 1 & 0 & & & & & & \\
x & x & 0 & 1 & & & & & & \\
y & y & & & x & x & & & & \\
y & y & & & x & x & & & & \\
 & & & & x & x & 1 & 0 & & \\
 & & & & x & x & 0 & 1 & & \\
 & & & & y & y & & & x & x \\
 & & & & y & y & & & x & x
\end{pmatrix}
\begin{pmatrix}
\delta u_1 \\ \delta v_1 \\ \delta u_2 \\ \delta v_2 \\ \delta u_3 \\ \delta v_3 \\ \delta u_4 \\ \delta v_4 \\ \delta u_5 \\ \delta v_5
\end{pmatrix}
=
\begin{pmatrix}
x \\ x \\ x \\ x \\ x \\ x \\ x \\ x
\end{pmatrix}, \qquad (4.31)
$$

where the "y"s are newly filled elements in the system. Now, we eliminate δu_3 and δv_3 from Equations (IV'), using Equations (II') and full pivoting over columns 5 and

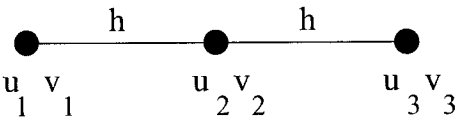


Figure 4.3: Two adjacent 2D elements

6 of Equations (II') and (IV'). As a result, System (4.31) changes to the form:

$$\begin{array}{l} I''a) \\ I''b) \\ II''a) \\ II''b) \\ III''a) \\ III''b) \\ IV''a) \\ IV''b) \end{array} \begin{pmatrix} x & x & 1 & & & & & & \\ x & x & & 1 & & & & & \\ y & y & & & 1 & 0 & & & \\ y & y & & & 0 & 1 & & & \\ & & & & x & x & 1 & & \\ & & & & x & x & & 1 & \\ z & z & & & & & & x & x \\ z & z & & & & & & x & x \end{pmatrix} \begin{pmatrix} \delta u_1 \\ \delta v_1 \\ \delta u_2 \\ \delta v_2 \\ \delta u_3 \\ \delta v_3 \\ \delta u_4 \\ \delta v_4 \\ \delta u_5 \\ \delta v_5 \end{pmatrix} = \begin{pmatrix} x \\ x \\ x \\ x \\ x \\ x \\ x \\ x \end{pmatrix}. \tag{4.32}$$

Here the "z"s are newly filled elements in the system. Equations (IV'') can be solved, after adding the global boundary conditions for the whole domain. As one can see, the stability of this procedure depends on the elimination of common variables in two adjacent regions. We must be able to determine Equations (4.20), using the elimination process with the full pivoting strategy, for otherwise we cannot express δu and δv in terms of their neighboring values.

Here, we consider a simple example, in order to see the result of the procedure for two adjacent elements, as shown in Figure 4.3, (This is the same example as in Section 3.10.1). The system for these two elements is

$$1/h \begin{pmatrix} 1 & h & -1 & 0 & & \\ 1 & 0 & -1 & h & & \\ & & 1 & h & -1 & 0 \\ & & 1 & 0 & -1 & h \end{pmatrix} \begin{pmatrix} \delta u_1 \\ \delta v_1 \\ \delta u_2 \\ \delta v_2 \\ \delta u_3 \\ \delta v_3 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix}, \tag{4.33}$$

where $c_1, \cdots, c_4$ are the right hand side coefficients. After elimination of u_3 and v_3 we have

$$\begin{pmatrix} -1 & -h & 1 & 0 & & \\ 0 & -1 & 0 & 1 & & \\ 1/h & 2 & & & -1/h & 0 \\ 1/h & 1 & & & -1/h & 1 \end{pmatrix} \begin{pmatrix} \delta u_1 \\ \delta v_1 \\ \delta u_2 \\ \delta v_2 \\ \delta u_3 \\ \delta v_3 \end{pmatrix} = \begin{pmatrix} c_1' \\ c_2' \\ c_3' \\ c_4' \end{pmatrix}. \tag{4.34}$$

The two last equations of the system (4.34) can be easily solved with proper boundary conditions.

4.6.2 Example

Reconsider the special example in the preceding chapter, namely, Example 3.10.2, for which we have shown that the original nested dissection method, as presented in the previous chapter fails. Consider the second order scalar linear ODE of the previous chapter

$$Lu = u''(x) + \lambda u(x) = f(x), \qquad 0 \le x \le 1. \tag{4.35}$$

with boundary condition

$$u(0) = u(1) = 0. \tag{4.36}$$

Consider again an element, a line segment, as shown in Figure 3.7. It is centered at x_0, has a length h, and one collocation point (z) at the center of the segment. The matching points are at $x_0 - h/2$ and $x_0 + h/2$. As basis functions of the polynomial space $P_{m+2} = P_3$, we choose $\{1, x, x^2\}$.

The matrices A and B are obtained by solving System (3.11) or (4.11), which is

$$\begin{pmatrix} 1 & 1 & \lambda \\ x_0 - h/2 & x_0 + h/2 & \lambda x_0 \\ (x_0 - h/2)^2 & (x_0 + h/2)^2 & 2 + \lambda x_0^2 \end{pmatrix} \begin{pmatrix} \alpha_{11} & \alpha_{21} \\ \alpha_{12} & \alpha_{22} \\ \beta_1 & \beta_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 2x_0 - h & 2x_0 + h \end{pmatrix}.$$

Upon solving this system, we find:

$$A = \frac{1}{\lambda h^3 - 8h} \begin{pmatrix} 8 - 3\lambda h^2 & -8 - \lambda h^2 \\ 8 + 3\lambda h^2 & -8 + 3\lambda h^2 \end{pmatrix}, \qquad B = \frac{4h}{\lambda h^2 - 8} \begin{pmatrix} 1 \\ -1 \end{pmatrix}. \tag{4.37}$$

Take $f(x) \equiv 0$. Using the modified nested dissection method, we have:

$$\gamma \begin{pmatrix} -1/\gamma & 8-3l & 0 & -8-l & 0 & 0 \\ 0 & 8+3l & -1/\gamma & -8+3l & 0 & 0 \\ 0 & 0 & -1/\gamma & 8-3l & 0 & -8-l \\ 0 & 0 & 0 & 8+3l & -1/\gamma & -8+3l \end{pmatrix} \begin{pmatrix} v_1 \\ u_1 \\ v_2 \\ u_2 \\ v_3 \\ u_3 \end{pmatrix} = 0, \tag{4.38}$$

where $\gamma = 1/(\lambda h^3 - 8h)$ and $l = \lambda h^2$. If we take $\lambda = 8/(3h^2)$ then the value of E in Equation (3.38) is zero, and as a result, we cannot solve this system using the original nested dissection in Section 3.8. Now, System (4.38) with $\lambda = 8/(3h^2)$ is given by:

$$\frac{3}{16h} \begin{pmatrix} 16h/3 & 0 & 0 & 32/3 & 0 & 0 \\ 0 & -32/3 & 16h/3 & 0 & 0 & 0 \\ 0 & 0 & 16h/3 & 0 & 0 & 32/3 \\ 0 & 0 & 0 & -32/3 & 16h/3 & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ u_1 \\ v_2 \\ u_2 \\ v_3 \\ u_3 \end{pmatrix} = 0. \tag{4.39}$$

Eliminating of v_2 and u_2 from the first and the second lines of System (4.39), using the third and the forth lines of this system, and full pivoting, we obtain the following system:

$$\frac{3}{16h} \begin{pmatrix} 16h/3 & 0 & 0 & 0 & 16h/3 & 0 \\ 0 & -32/3 & 0 & 0 & 0 & -32/3 \\ 0 & 0 & 16h/3 & 0 & 0 & 32/3 \\ 0 & 0 & 0 & -32/3 & 16h/3 & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ u_1 \\ v_2 \\ u_2 \\ v_3 \\ u_3 \end{pmatrix} = 0. \tag{4.40}$$

Using the first and the second lines of the above system, we can write

$$\frac{3}{16h} \begin{pmatrix} 16h/3 & 0 & 16h/3 & 0 \\ 0 & -32/3 & 0 & -32/3 \end{pmatrix} \begin{pmatrix} v_1 \\ u_1 \\ v_3 \\ u_3 \end{pmatrix} = 0. \tag{4.41}$$

This system is again of the form Equation (4.23), but for the enlarged region (two adjacent elements together). We see that the modified nested dissection algorithm does not fail for this example, in contrast to the original algorithm.

Chapter 5

The Collocation Method for Linear Elliptic PDE BVP

5.1 Introduction

In this chapter, we consider algorithmic aspects of a class of finite element collocation methods for the approximate numerical solution of linear PDEs. As mentioned in Section 3.1, the finite element collocation method with discontinuous piecewise polynomials can be used for the solution of ODE BVPs as well as PDEs. One of the advantages of the method is, in fact, that it can be used for a wide range of problems. It can be considered as a unified method for solving ODE BVPs and elliptic PDEs. In this chapter, we extend the method described in Chapter 3 for ODEs to PDEs.

Important characteristics of the collocation method for PDEs are:

- high order of accuracy is possible,
- the piecewise polynomial solutions are not globally continuous,

- the linear systems that arise can be solved efficiently by the method of nested dissection,
  and
- the possibility to use different types of elements, including rectangles and triangles.

In this thesis, we consider only rectangular elements. Locally, for each finite element, the approximate solution is a polynomial. Although polynomials corresponding to adjacent elements don't need to match continuously, their values and normal derivatives must match at a discrete set of points on the common boundary. High order accuracy can be attained by increasing the number of matching points and the number of collocation points for each finite element. As in Chapter 3, collocation methods can be defined equivalently as generalized finite difference methods. The linear equations that arise from the discretisation lend themselves well to solution by the method of nested dissection.

5.2 Collocation with discontinuous piecewise polynomials

Consider the following second order linear elliptic PDE

$$Lu = \Delta u + a(x)^T \nabla u + b(x)u = f(x), \qquad x \in \Omega \subset R^2 \tag{5.1}$$

where Δ is Laplace's operator, ∇u is the gradient of u, $a(x) \in R^2$, $b(x), f(x), u(x) \in R$, and ^{T} denotes transpose. As boundary condition take

$$u(x) = u_0(x), \qquad x \in \delta\Omega. \tag{5.2}$$

We shall assume throughout that the coefficient functions a and b, and the inhomogeneous term f are sufficiently smooth. We shall also assume that Equations (5.1) and (5.2) has a unique solution. Consider the domain Ω, for simplicity a square[1], in Figure 5.1a. Initially Ω is subdivided into two subdomains or *regions*, Figure 5.1b. Thereafter, each of these regions is subdivided again into two parts, as in Figure

[1]The domain Ω can in principle be quite general. It is also possible to treat elliptic systems with more general boundary conditions than we study in this section.

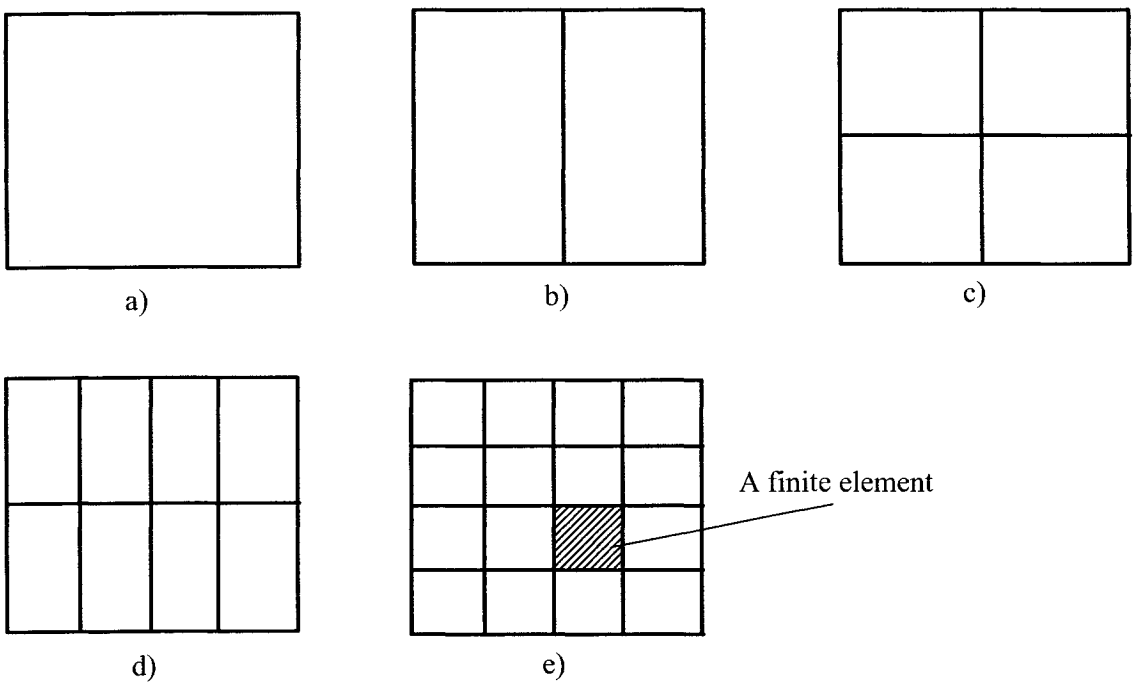


Figure 5.1: Recursive subdivision

5.1c. This procedure is continued recursively until a desired level of refinement is achieved, Figure 5.1e. The smallest regions are the *finite elements*. These elements are leaf nodes in the binary recursion tree. The recursion is not really necessary for the discretisation, but it is useful for the nested dissection algorithm.

5.3 The finite difference formulation

Intuitively, for any finite element, for example, Ω_ℓ, if $u(x)$ is known on $\delta\Omega_\ell$ and if Ω_ℓ is sufficiently small then the solution $u(x)$ is defined in Ω_ℓ. As a result, $\partial u/\partial\eta$, the derivative of u in the direction of the outward normal η to the boundary $\delta\Omega_\ell$, will be defined along any smooth part of the boundary. This relation defines a mapping from $\delta\Omega_\ell$ to $\delta\Omega_\ell$. This mapping is linear in u and f. The finite difference method is based on a discrete version of this basic observation. For any finite element Ω_ℓ (refer to Figure 5.2), the above idea leads to the following finite difference discretisation formula for Equation (5.1):

$$v_i = \sum_{j=1}^{n} \alpha_{ij} u_j + \sum_{j=1}^{m} \beta_{ij} f(z_j), \qquad i = 1, \cdots, n. \tag{5.3}$$

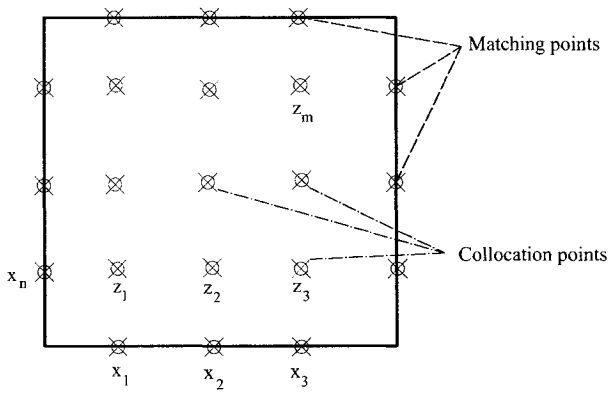


Figure 5.2: A finite element of Ω

Here m is the number of collocation points, n is the number of matching points, and u_i denote the approximate solution at point x_i, $(i = 1, \cdots, n)$ on the boundary of the finite element, that is, $u_i \approx u(x_i)$. Similarly v_i denotes the normal derivative at x_i, *i.e.*, $v_i \approx \nabla u(x_i)^T \eta_i$, where η_i is the unit exterior normal to the finite element boundary at x_i. The points z_j lie normally inside Ω_ℓ.

Thus, for each finite element of Ω_ℓ there is a discrete equation corresponding to each x_i on the boundary of the finite element. The coefficients α_{ij} and β_{ij} in Equation (5.3) are determined by requiring (5.3) to be satisfied exactly for a basic set of equations with known solutions. These test problems are of the form (5.1), with $f = L\phi$ where ϕ is a polynomial, and with obvious solution $u = \phi$. More precisely, let P_{n+m} be a polynomial space of dimension $n + m$. Then we require that

$$\nabla\phi(x_i)^T\eta_i = \sum_{j=1}^{n} \alpha_{ij}\phi(x_j) + \sum_{j=1}^{m} \beta_{ij} L\phi(z_j), \qquad i = 1, \cdots, n, \quad \forall \phi \in P_{n+m}. \tag{5.4}$$

Let $u = (u_1, \cdots, u_n)^T$, $v = (v_1, \cdots, v_n)^T$, $f = (f(z_1), \cdots, f(z_m))^T$, and

$$A = \begin{pmatrix} \alpha_{11} & \cdots & \alpha_{1n} \\ \vdots & & \vdots \\ \alpha_{n1} & \cdots & \alpha_{nn} \end{pmatrix}, \qquad B = \begin{pmatrix} \beta_{11} & \cdots & \beta_{1m} \\ \vdots & & \vdots \\ \beta_{n1} & \cdots & \beta_{nm} \end{pmatrix}.$$

Then Equation (5.3) can be written as

$$v = Au + Bf. \tag{5.5}$$

Furthermore, if one defines

$$\Phi = \begin{pmatrix} \phi_1(x_1) & \cdots & \phi_1(x_n) \\ \vdots & & \vdots \\ \phi_{n+m}(x_1) & \cdots & \phi_{n+m}(x_n) \end{pmatrix}, \tag{5.6}$$

$$L_\Phi = \begin{pmatrix} L\phi_1(z_1) & \cdots & L\phi_1(z_m) \\ \vdots & & \vdots \\ L\phi_{n+m}(z_1) & \cdots & L\phi_{n+m}(z_m) \end{pmatrix} \tag{5.7}$$

and

$$R_\Phi = \begin{pmatrix} \nabla\phi_1(x_1)^T\eta_1 & \cdots & \nabla\phi_1(x_n)^T\eta_n \\ \vdots & & \vdots \\ \nabla\phi_{n+m}(x_1)^T\eta_1 & \cdots & \nabla\phi_{n+m}(x_n)^T\eta_n \end{pmatrix}, \tag{5.8}$$

where $Span\{\phi_1, \cdots, \phi_{n+m}\} = P_{n+m}$, then Equations (5.4) can be written as

$$(\Phi \mid L_\Phi) \begin{pmatrix} A^T \\ B^T \end{pmatrix} = R_\Phi. \tag{5.9}$$

As was the case for ODEs in Section 3.4, for the finite difference approximation to be well defined, the matrix $(\Phi \mid L_\Phi)$ must be nonsingular. Although this condition is not automatically satisfied, there are many possible schemes for choosing the x_i, z_j, and the space P_{n+m}, so that it is satisfied. Two such schemes are given in Section 5.6.1. Note that if the coefficient functions $a(x)$ and $b(x)$ in Equation (5.1) are constant and if all finite elements are identical then Equation (5.9) needs only be solved once. This observation is further developed in Section 5.6.1.

Of course, in addition to the finite difference Equations (5.3), boundary conditions must be satisfied, *i.e.*,

$$u_i = u_0(x_i) \text{ at points } x_i \text{ that lie on } \delta\Omega. \tag{5.10}$$

5.4 The collocation formulation

If $(\Phi \mid L_\Phi)$ is nonsingular for each finite element, then the finite difference scheme is equivalent to the following collocation scheme (*cf.* [45]). Associate a polynomial $p(x) \in P_{n+m}$ to each finite element. For any two adjacent finite elements, require that at each point x_i on the common boundary:

$$\begin{aligned} &\triangleright \text{ the values of neighboring polynomials match,} \\ &\quad \text{and} \\ &\triangleright \text{ the normal derivatives of neighboring polynomials match.} \end{aligned} \tag{5.11}$$

Furthermore, each polynomial must satisfy the collocation equations

$$Lp(z_k) = f(z_k), \qquad k = 1, \cdots, m, \tag{5.12}$$

at all collocation points z_k of the corresponding finite element. Finally, the boundary conditions must be satisfied, *i.e.*, for each boundary adjacent element we have

$$p(x_i) = u_0(x_i) \text{ at the points } x_i \text{ that lie on } \delta\Omega. \tag{5.13}$$

The *equivalence* of the finite difference scheme and the collocation scheme is presented in the following theorem.

Theorem: 5.4.1 *Assume for any finite element Ω_ℓ the matrix $(\Phi \mid L_\Phi)$ is nonsingular and the matrices A and B are defined by Equation (5.9). If we find a solution of the collocation scheme, Equations (5.11) to (5.13), and if for each finite element, we evaluate the local collocation polynomial at the points x_i then the resulting values satisfy the finite difference Equations (5.5).*

Conversely, if we have a solution to the system of all finite difference Equations (5.5) and boundary conditions (5.10), then, for each element, we can locally interpolate the finite difference solution by a polynomial $p(x) \in P_{n+m}$ that satisfies the differential equation at the collocation points. Furthermore, this polynomial satisfies the continuity conditions in (5.11).

Proof : *Take $p(x) = \sum_{i=1}^{n+m} c_i \phi_i(x)$. Then the collocation Equations (5.12) can be written as*

$$\sum_{i=1}^{n+m} c_i L\phi_i(z_k) = f(z_k).$$

Using the definition of L_Φ, we have

$$L_\Phi^T c = f,$$

where $c = (c_1, \cdots, c_{n+m})^T$. Take

$$\begin{aligned} u &= \Phi^T c, \\ v &= R_\Phi^T c. \end{aligned}$$

Then the pair (u, v) satisfies the finite difference Equation (5.5), because

$$v - Au - Bf = R_\Phi^T c - A\Phi^T c - Bf$$

$$
\begin{aligned}
&= [c^T(R_\Phi - \Phi A^T)]^T - Bf \\
&= [c^T(L_\Phi B^T)]^T - Bf \quad (\textit{using Equation (5.9)}) \\
&= BL_\Phi^T c - Bf = B(L_\Phi^T c - f) = 0.
\end{aligned}
$$

Conversely, let (u, v) be a solution of the finite difference Equations (5.5). We can find c by solving the following system

$$
\begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix} c = \begin{pmatrix} u \\ f \end{pmatrix}.
$$

Take

$$
p(x) = \sum_{i=1}^{n+m} c_i \phi_i(x).
$$

As a result, the vector c satisfies $\Phi^T c = u$, and the collocation equations, as we have

$$
L_\Phi^T c = f. \tag{5.14}
$$

Neighboring polynomials are also continuous at the points x_i, because

$$
(p(x_1), \cdots, p(x_n))^T = \Phi^T c = u.
$$

To prove the continuity of $\nabla p^T \eta$ note that

$$
\begin{aligned}
\begin{pmatrix} \nabla p(x_1)^T \eta_1 \\ \vdots \\ \nabla p(x_n)^T \eta_n \end{pmatrix} &= R_\Phi^T c \\
&= (\Phi A^T + L_\Phi B^T)^T c \quad (\textit{using Equation (5.9)}) \\
&= A\Phi^T c + BL_\Phi^T c \\
&= Au + Bf = v. \qquad (\textit{Equation (5.5)}
\end{aligned}
$$

5.5 Example

In this example, we choose the Laplace operator as the differential operator, $L = \Delta$. A square element as in Figure 5.3 is considered. Assume this element is centered at $(0, 0)$, and each side has the length $2h$. We take 4 matching points

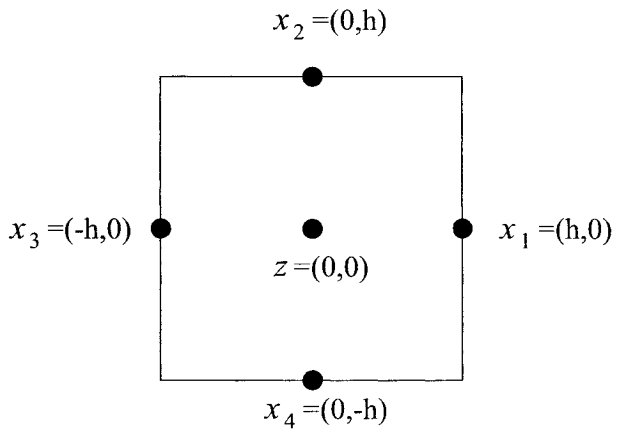


Figure 5.3: A finite element with one collocation point

($n = 4$), and we assume that these matching points are located at the midpoints of the four edges. One collocation point (z) is considered ($m = 1$), and we choose it at the center of the square element. For each matching point x_i the finite difference approximation (5.3) takes the form

$$v_i = \sum_{j=1}^{4} \alpha_{ij} u_j \; + \; \beta_i f(z).$$

As a basis of $P_{n+m} = P_5$, we choose the polynomials $\{1, x, y, x^2, y^2\}$. The coefficients α_{ij} and β_i are obtained by solving Equation (5.9), which are

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ h & 0 & -h & 0 & 0 \\ 0 & h & 0 & -h & 0 \\ h^2 & 0 & h^2 & 0 & 2 \\ 0 & h^2 & 0 & h^2 & 2 \end{pmatrix} \begin{pmatrix} \alpha_{11} & \alpha_{21} & \alpha_{31} & \alpha_{41} \\ \alpha_{12} & \alpha_{22} & \alpha_{32} & \alpha_{42} \\ \alpha_{13} & \alpha_{23} & \alpha_{33} & \alpha_{43} \\ \alpha_{14} & \alpha_{24} & \alpha_{34} & \alpha_{44} \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 2h & 0 & 2h & 0 \\ 0 & 2h & 0 & 2h \end{pmatrix}.$$

Solving this system we find that the finite difference equations are:

$$\begin{aligned} v_1 &= (2u_1 - u_2 - u_4)/2h + hf(z)/2, \\ v_2 &= (2u_2 - u_1 - u_3)/2h + hf(z)/2, \\ v_3 &= (2u_3 - u_2 - u_4)/2h + hf(z)/2, \\ v_4 &= (2u_4 - u_1 - u_3)/2h + hf(z)/2. \end{aligned}$$

5.6 Special basis functions

5.6.1 Selection of the basis functions

For each finite element, System (5.9) must be solved for the coefficients α_{ij} and β_{ij}. As in Section 3.11.1 for ODEs, this procedure can be performed efficiently for PDEs, if we divide the basis of P_{n+m} into two groups, namely $\{\phi_1, \cdots, \phi_n\}$ and $\{\psi_1, \cdots, \psi_m\}$. The matrices Φ, L_Φ and R_Φ in Section 3.11.1 (Equations (5.6)-(5.8)) can be rewritten as

$$\Phi = \begin{pmatrix} \tilde{\Phi} \\ \Psi \end{pmatrix} = \begin{pmatrix} \phi_1(x_1) & \cdots & \phi_1(x_n) \\ \vdots & & \vdots \\ \phi_n(x_1) & \cdots & \phi_n(x_n) \\ \psi_1(x_1) & \cdots & \psi_1(x_n) \\ \vdots & & \vdots \\ \psi_m(x_1) & \cdots & \psi_m(x_n) \end{pmatrix},$$

$$L_\Phi = \begin{pmatrix} L_{\tilde{\Phi}} \\ L_\Psi \end{pmatrix} = \begin{pmatrix} L\phi_1(z_1) & \cdots & L\phi_1(z_m) \\ \vdots & & \vdots \\ L\phi_n(z_1) & \cdots & L\phi_n(z_m) \\ L\psi_1(z_1) & \cdots & L\psi_1(z_m) \\ \vdots & & \vdots \\ L\psi_m(z_1) & \cdots & L\psi_m(z_m) \end{pmatrix},$$

and

$$R_\Phi = \begin{pmatrix} R_{\tilde{\Phi}} \\ R_\Psi \end{pmatrix} = \begin{pmatrix} \nabla\phi_1(x_1)^T\eta_1 & \cdots & \nabla\phi_1(x_n)^T\eta_n \\ \vdots & & \vdots \\ \nabla\phi_n(x_1)^T\eta_1 & \cdots & \nabla\phi_n(x_n)^T\eta_n \\ \nabla\psi_1(x_1)^T\eta_1 & \cdots & \nabla\psi_1(x_n)^T\eta_n \\ \vdots & & \vdots \\ \nabla\psi_m(x_1)^T\eta_1 & \cdots & \nabla\psi_m(x_n)^T\eta_n \end{pmatrix}.$$

From now on, we omit the $\tilde{}$ for notational simplicity. Equation (5.9) takes the following form

$$\begin{pmatrix} \Phi & L_\Phi \\ \Psi & L_\Psi \end{pmatrix} \begin{pmatrix} A^T \\ B^T \end{pmatrix} = \begin{pmatrix} R_\Phi \\ R_\Psi \end{pmatrix}. \tag{5.15}$$

Let Δ_Φ and Δ_Ψ represent the matrices L_Φ and L_Ψ, respectively, for the special case where $L = \Delta$, the Laplace operator. As in the case of ODEs (Section 3.11.1), for many particular PDEs, one can construct the basis such that it satisfies the following equation

$$\begin{pmatrix} \Phi & \Delta_\Phi \\ \Psi & \Delta_\Psi \end{pmatrix} = \begin{pmatrix} I & O \\ O & I \end{pmatrix}. \tag{5.16}$$

In such cases, where $L = \Delta$, the equation (5.15) has the explicit solution

$$A = R_\Phi^T,$$

$$B = R_\Psi^T.$$

Selecting the same special basis for the case of the more general operator L in Equation (5.1), System (5.15) can be written as

$$\begin{pmatrix} I & L_\Phi \\ O & L_\Psi \end{pmatrix} \begin{pmatrix} A^T \\ B^T \end{pmatrix} = \begin{pmatrix} R_\Phi \\ R_\Psi \end{pmatrix}.$$

This system can be solved efficiently in two following steps

$$\begin{aligned} i) \quad & L_\Psi B^T = R_\Psi, \\ ii) \quad & A^T = R_\Phi - L_\Phi B^T. \end{aligned} \tag{5.17}$$

The entries in L_Φ and L_Ψ also simplify because $\Delta\phi_i(z_j) = 0$ and $\Delta\psi_i(z_j) = \delta_{ij}$. Thus

$$\begin{aligned} [L_\Phi]_{ij} = L\phi_i(z_j) = & \quad a(z_j)^T \nabla\phi_i(z_j) + b(z_j)\phi_i(z_j), \\ [L_\Psi]_{ij} = L\psi_i(z_j) = & \quad \delta_{ij} + a(z_j)^T \nabla\psi_i(z_j) + b(z_j)\psi_i(z_j). \end{aligned} \tag{5.18}$$

Above δ_{ij} denotes the Kronecker delta function.

5.6.2 Evaluation of the weights

Using the special basis, for each finite element the coefficients α_{ij} and β_{ij}, *i.e.*, the matrices A and B, can be obtained from System (5.17). As a result, we must evaluate L_Φ, L_Ψ, R_Φ and R_Ψ, *i.e.*, using Equations (5.18), we need the values of ϕ_i, $\nabla\phi_i$, ψ_i, and $\nabla\psi_i$ at the collocation points and the values of $\nabla\phi_i$ and $\nabla\psi_i$ at the matching points. These quantities, as in Section 3.11.2, are called *weights*. Note that, as in the ODE case, second derivatives are not needed. For the uniform discretization (mesh) each finite element has the same set of weights. Even if the discretization is not uniform then the weights of similar finite elements are related by a scaling factor. In these cases, the weights can be pre-computed. In other cases, when elements are not similar, it may not be beneficial to pre-compute the weights or even to use the special basis, and one would compute the α_{ij} and β_{ij} using Equation (5.9).

For completeness we now show in detail how to pre-compute the weights $\phi_i(z_j)$ and $\psi_i(z_j)$. The procedure is similar to that for ODEs (Section 3.11.2). Let θ_i, $i = 1, \cdots, n+m$ denote a monomial basis of P_{n+m}. For example, one can take $P_5 = Span\{1, x, y, x^2, y^2\}$. Monomials and their derivatives can be evaluated efficiently. We have

$$\phi_i(x) = \sum_{k=1}^{n+m} c_{ik}\theta_k(x) \quad \text{and} \quad \psi_i(x) = \sum_{k=1}^{n+m} d_{ik}\theta_k(x),$$

To satisfy Equation (5.16), we must have

$$\begin{aligned} \phi_i(x_j) &= \delta_{ij}, & \Delta\phi_i(z_j) &= 0, \\ \psi_i(x_j) &= 0, & \Delta\psi_i(z_j) &= \delta_{ij}, \end{aligned}$$

that is,

$$\begin{aligned} \sum_{k=1}^{n+m} c_{ik}\theta_k(x_j) &= \delta_{ij}, & \textstyle\sum_{k=1}^{n+m} c_{ik}\Delta\theta_k(z_j) &= 0, \\ \sum_{k=1}^{n+m} d_{ik}\theta_k(x_j) &= 0, & \textstyle\sum_{k=1}^{n+m} d_{ik}\Delta\theta_k(z_j) &= \delta_{ij}. \end{aligned} \tag{5.19}$$

Define

$$\Theta = \begin{pmatrix} \theta_1(x_1) & \cdots & \theta_1(x_n) \\ \vdots & & \vdots \\ \theta_{n+m}(x_1) & \cdots & \theta_{n+m}(x_n) \end{pmatrix}, \qquad \Delta_\Theta = \begin{pmatrix} \Delta\theta_1(z_1) & \cdots & \Delta\theta_1(z_m) \\ \vdots & & \vdots \\ \Delta\theta_{n+m}(z_1) & \cdots & \Delta\theta_{n+m}(z_m) \end{pmatrix},$$

and

$$C = \begin{pmatrix} c_{11} & \cdots & c_{1,n+m} \\ \vdots & & \vdots \\ c_{n1} & \cdots & c_{n,n+m} \end{pmatrix}, \qquad D = \begin{pmatrix} d_{11} & \cdots & d_{1,n+m} \\ \vdots & & \vdots \\ d_{m1} & \cdots & d_{m,n+m} \end{pmatrix},$$

then Equation (5.19) can be written as

$$\begin{pmatrix} \Theta^T \\ \Delta_\Theta^T \end{pmatrix} (C^T \mid D^T) = \begin{pmatrix} I & O \\ O & I \end{pmatrix}.$$

Thus

$$(C^T \mid D^T) = \begin{pmatrix} \Theta^T \\ \Delta_\Theta^T \end{pmatrix}^{-1}. \tag{5.20}$$

To evaluate $\phi_i(z_j)$ and $\psi_i(z_j)$ we have

$$\phi_i(z_j) = \sum_{k=1}^{n+m} c_{ik}\theta_k(z_j), \quad \text{and} \quad \psi_i(z_j) = \sum_{k=1}^{n+m} d_{ik}\theta_k(z_j), \tag{5.21}$$

Defining

$$\Phi_z = \begin{pmatrix} \phi_1(z_1) & \cdots & \phi_1(z_m) \\ \vdots & & \vdots \\ \phi_n(z_1) & \cdots & \phi_n(z_m) \end{pmatrix}, \qquad \Psi_z = \begin{pmatrix} \psi_1(z_1) & \cdots & \psi_1(z_m) \\ \vdots & & \vdots \\ \psi_m(z_1) & \cdots & \psi_m(z_m) \end{pmatrix}$$

and

$$\Theta_z = \begin{pmatrix} \theta_1(z_1) & \cdots & \theta_1(z_m) \\ \vdots & & \vdots \\ \theta_{n+m}(z_1) & \cdots & \theta_{n+m}(z_m) \end{pmatrix},$$

one can write Equation (5.21) as

$$\begin{pmatrix} \Phi_z \\ \Psi_z \end{pmatrix} = \begin{pmatrix} C \\ D \end{pmatrix} \Theta_z,$$

or

$$(\Phi_z^T \mid \Psi_z^T) = \Theta_z^T (C^T \mid D^T).$$

Now using Equation (5.20) it follows that

$$(\Phi_z^T \mid \Psi_z^T) = \Theta_z^T \begin{pmatrix} \Theta^T \\ \Delta_\Theta^T \end{pmatrix}^{-1},$$

from which

$$(\Phi_z^T \mid \Psi_z^T) \begin{pmatrix} \Theta^T \\ \Delta_\Theta^T \end{pmatrix} = \Theta_z^T,$$

or

$$(\Theta \mid \Delta_\Theta) \begin{pmatrix} \Phi_z \\ \Psi_z \end{pmatrix} = \Theta_z. \tag{5.22}$$

Thus the weights $\phi_i(z_j)$ and $\psi_i(z_j)$ can be computed by solving the $n+m$ by $n+m$ System (5.22) for Φ_z and Ψ_z.

The other weights, namely, $\nabla\phi_i$ and $\nabla\psi_i$ at the collocation points z_j and at the matching points x_j, can be computed similarly using the same $n+m$ by $n+m$ matrix. Thus only one LU-decomposition is needed. Furthermore, as mentioned earlier, the weights for similar finite elements can be obtained by scaling.

5.7 Nested dissection

As Equation (5.5) is similar to Equations (3.8) and (4.12), we can use the nested dissection procedure, described in Section 3.8 or Section 4.4.1 for ODEs, to eliminate the unknowns u and v on boundaries separating adjacent regions. As before, this procedure results in the elimination of all interior unknowns. One is left with one equation for each x_i on $\delta\Omega$. Thereafter, the boundary conditions can be used to determine the values of u and v at the points x_i on $\delta\Omega$. Finally, a recursive back-substitution gives the values of u (and hence v) at the matching points on each interior boundary.

In Figure (5.4), two adjacent regions Ω_1 and Ω_2 for a two-dimensional problem are shown. As before, the elimination of the unknowns u and v on the common

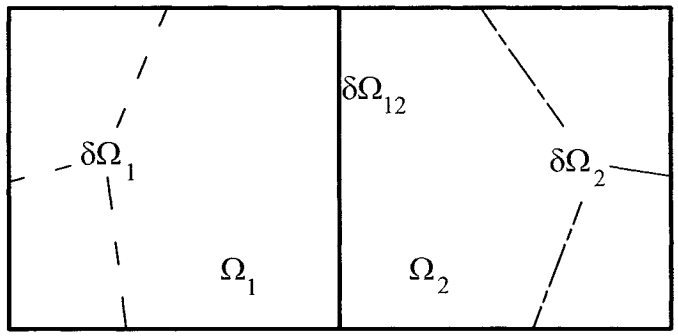


Figure 5.4: Two adjacent elements in a 2D domain

boundary is done as in domain decomposition. The common boundary is called $\delta\Omega_{12}$, and the remaining parts of the boundaries of Ω_1 and Ω_2 are called $\delta\Omega_1$ and $\delta\Omega_2$ respectively. (As one can see, in contrast to the case of ODEs, here the common boundary and other boundaries are line segments.) The vector u in the collocation formulation Equation (5.5) is split accordingly into u_{12} and u_1 for region Ω_1 and u_{21} $(= u_{12})$ and u_2 for region Ω_2. The vector v is split similarly. To prevent the problem mentioned in Section 3.10.1 (the original nested dissection procedure can fail due to possible singularity of the matrix E in Equation (3.37)), we use the modified procedure presented in Section 4.4.1. For this more flexible procedure, we rewrite Equation (5.5) as

$$Dv = Au + g, \tag{5.23}$$

where $g = Bf$ in Equation (5.5). As for the ODE case, the matrix D is the identity matrix at the element level but it is not necessarily an identity matrix at the interior nodes of the binary tree during the nested dissection. Thus for region Ω_1 Equations (5.5) can be written as

$$\begin{pmatrix} D^1_{11} & D^1_{12} \\ D^1_{21} & D^1_{22} \end{pmatrix} \begin{pmatrix} v_1 \\ v_{12} \end{pmatrix} = \begin{pmatrix} A^1_{11} & A^1_{12} \\ A^1_{21} & A^1_{22} \end{pmatrix} \begin{pmatrix} u_1 \\ u_{12} \end{pmatrix} + \begin{pmatrix} g^1_1 \\ g^1_{12} \end{pmatrix}, \tag{5.24}$$

and for Ω_2, we have

$$\begin{pmatrix} D^2_{11} & D^2_{12} \\ D^2_{21} & D^2_{22} \end{pmatrix} \begin{pmatrix} v_2 \\ v_{21} \end{pmatrix} = \begin{pmatrix} A^2_{11} & A^2_{12} \\ A^2_{21} & A^2_{22} \end{pmatrix} \begin{pmatrix} u_2 \\ u_{21} \end{pmatrix} + \begin{pmatrix} g^2_2 \\ g^2_{21} \end{pmatrix}. \tag{5.25}$$

Above $\begin{pmatrix} g^1_1 \\ g^1_{12} \end{pmatrix}$ and $\begin{pmatrix} g^2_2 \\ g^2_{21} \end{pmatrix}$ correspond to the term g for region Ω_1 and Ω_2 respectively. The superscript represents the region number. From the continuity relations (5.11),

we can write

$$\begin{aligned} u_{12} &= u_{21}, \\ v_{12} &= -v_{21}. \end{aligned} \tag{5.26}$$

Since v is the outward normal to the boundary at the matching point, the direction of v_{12} is opposite to the direction of v_{21}. As result, we have $v_{12} = -v_{21}$. The full system for two adjacent regions can, as in Section 4.4.1, be written as

$$\begin{pmatrix} D_{22}^1 & -A_{22}^1 & D_{21}^1 & -A_{21}^1 & 0 & 0 \\ -D_{22}^2 & -A_{22}^2 & 0 & 0 & D_{21}^2 & -A_{21}^2 \\ D_{12}^1 & -A_{12}^1 & D_{11}^1 & -A_{11}^1 & 0 & 0 \\ -D_{12}^2 & -A_{12}^2 & 0 & 0 & D_{11}^2 & -A_{11}^2 \end{pmatrix} \begin{pmatrix} v_{12} \\ u_{12} \\ v_1 \\ u_1 \\ v_2 \\ u_2 \end{pmatrix} = \begin{pmatrix} g_{12}^1 \\ g_{21}^2 \\ g_1^1 \\ g_2^2 \end{pmatrix}. \tag{5.27}$$

Using the full pivoting strategy described in Section 4.4.1, we obtain the following system:

$$\begin{pmatrix} I & 0 & \hat{D}_{11} & \hat{A}_{11} & \hat{D}_{12} & \hat{A}_{12} \\ 0 & I & \hat{D}_{21} & \hat{A}_{21} & \hat{D}_{22} & \hat{A}_{22} \\ 0 & 0 & D_{11} & C_{11} & D_{12} & C_{12} \\ 0 & 0 & D_{21} & C_{21} & D_{22} & C_{22} \end{pmatrix} \begin{pmatrix} v_{12} \\ u_{12} \\ v_1 \\ u_1 \\ v_2 \\ u_2 \end{pmatrix} = \begin{pmatrix} \hat{g}_{12} \\ \hat{g}_{21} \\ \hat{g}_1 \\ \hat{g}_2 \end{pmatrix}. \tag{5.28}$$

From Equation (5.28), we can write

$$u_{12} = \hat{g}_{21} - (\hat{D}_{21}v_1 + \hat{A}_{21}u_1 + \hat{D}_{22}v_2 + \hat{A}_{22}u_2), \tag{5.29}$$

$$v_{12} = \hat{g}_{12} - (\hat{D}_{11}v_1 + \hat{A}_{11}u_1 + \hat{D}_{12}v_2 + \hat{A}_{12}u_2), \tag{5.30}$$

and

$$\begin{pmatrix} D_{11} & C_{11} & D_{12} & C_{12} \\ D_{21} & C_{21} & D_{22} & C_{22} \end{pmatrix} \begin{pmatrix} v_1 \\ u_1 \\ v_2 \\ u_2 \end{pmatrix} = \begin{pmatrix} \hat{g}_1 \\ \hat{g}_2 \end{pmatrix}. \tag{5.31}$$

Equation (5.31) represents the discrete equations for the enlarged region, that is, for the union of Ω_1 and Ω_2, after the elimination of the common boundary unknowns

u_{12} and v_{12}. It is important to note that these new equations are again of the form Equation (5.23).

The number of arithmetic operations required by the above nested dissection method can be shown to be $\mathcal{O}(N^3)$ for a N by N mesh (with N^2 elements). This compares favorably with the $\mathcal{O}(N^4)$ required by a standard band-solver. See [45] for more details.

5.8 Summary of the complete collocation algorithm

The algorithm of the finite element collocation method for linear PDEs is similar to the algorithm presented in Section 3.10 for ODEs:

- Mesh generation of 2D domain:
  - Recursive subdivision of the domain in x and y direction until a desired mesh size is reached.
  - Construction of a binary tree to be used in the nested dissection procedure.
- Loop over all elements
  - Compute the matrices Φ, L_Φ, and R_Φ using Equations (5.6), (5.7), and (5.8).
  - Solve Equation (5.9) to find A and B for each element.
- Determine Equation (5.5) for the root node of the binary tree, using the modified nested dissection procedure described in Section 5.7.
- Use the boundary conditions, together with Equation (5.5) for the root node, to solve for the unknowns on the boundary of the full domain Ω.
- Determine the unknowns at the interior mesh points using the recursive back-substitution, using Equations (5.29) and (5.30).

As one can see, we follow the same steps as in the one-dimensional case, but here we apply them to a two-dimensional problem.

5.9 Stability of the solution algorithm

As before for the ODE case, the stability of the solution algorithm depends on the stability of the modified nested dissection method in the above algorithm. We have used the same nested dissection method as Section 4.6.1, and therefore the same arguments that we used in Section 4.6.1 are applicable here.

5.10 Linear algebra stability considerations

In this section, we consider certain aspects of the stability of the collocation method with discontinuous piecewise polynomials for solving linear PDEs, using linear algebra. This type of investigation has also been considered by Sun and Wu [143]. To be able to consider our method in general form, *i.e.*, without considering the solution procedure, we write Equation (5.5) for each finite element mesh and we consider the solvability of the entire system for all "u"s and "v"s at the matching points. This system is much bigger than the final system that arises from the nested dissection solution procedure. If such system is stable, one can conclude that the method is stable. (Of course, the solution procedure could still be unstable).

The second line of Equations (5.24) and (5.25) can be written as

$$v_{12} = A^1_{21}u_1 + A^1_{22}u_{12} + g^1_{12}, \tag{5.32}$$

and

$$v_{21} = A^2_{21}u_2 + A^2_{22}u_{21} + g^2_{21}. \tag{5.33}$$

Note that D^1_{21} and D^2_{21} are the Null matrices, and D^1_{22} and D^2_{22} are the identity matrices at the element level. To have a final system only in terms of "u"s, we eliminate "v" at each matching point between two adjacent elements, and combine Equations (5.32), (5.33), and (5.26) to find the following equation:

$$A^1_{21}u_1 + A^2_{21}u_2 + (A^2_{22} + A^1_{22})u_{12} + g^1_{12} + g^2_{21} = 0. \tag{5.34}$$

Note that as v_{12} is equal to $-v_{21}$ we have $v_{12} + v_{21} = 0$, which yields an equation only in term of the "u"s. Every common boundary is between two adjacent elements, either on vertical edges (left and right) or horizontal edges (top and bottom). If we

collect the above equation for all the matching points, without considering how we construct the binary recursion tree, together with the domain boundary conditions, then we obtain a square system, to be solved for all "u"s.

As an example, consider the following PDE

$$Lu = \Delta u = f(x), \qquad x \in \Omega \subset R^2, \tag{5.35}$$

where Δ is the Laplace operator and $f(x), u(x) \in R$. As boundary condition we take

$$u(x) = 0, \qquad x \in \delta\Omega. \tag{5.36}$$

If we write Equation (5.5) for a square finite element, having sides of length "h", with one collocation point (similar to Figure 5.3), we find

$$A = \begin{bmatrix} 2h^{-1} & -h^{-1} & 0 & -h^{-1} \\ -h^{-1} & 2h^{-1} & -h^{-1} & 0 \\ 0 & -h^{-1} & 2h^{-1} & -h^{-1} \\ -h^{-1} & 0 & -h^{-1} & 2h^{-1} \end{bmatrix}. \tag{5.37}$$

One fact about this matrix is that for any element a_{ij}, $i \neq j$, we have $a_{ij} \leq 0$, $a_{ii} > 0$, and the sum of off-diagonal terms is exactly equal to the negative value of the diagonal term,

$$a_{ii} = -\sum_{j=1, j\neq i}^{n} a_{ij}. \tag{5.38}$$

We have, $n = 4$ for the matrix of Equation (5.37). Such a matrix is called diagonally dominant, which is nonsingular if it is also irreducible.

Consider two elements, corresponding to a 2 by 1 mesh in Figure 5.5, with one collocation point. For the matching point between these two elements, Equation (5.34) can be written as

$$\sum_{i=1, i\neq k}^{4} a_i u_i + \sum_{j=1, j\neq l}^{4} a_j' u_{j+4} + (a_k + a_l')u_{12} = -g_{12}^1 - g_{21}^2, \tag{5.39}$$

where, a_i, $i = 1, \cdots, 4$, (line 4 of the matrix A in Equation (5.37)) are the coefficients of $u_1, \cdots, u_4$ of the first element, a_j', $j = 1, \cdots, 4$, (line 2 of the matrix A in Equation (5.37)) are the coefficients of $u_5, \cdots, u_8$ of the second element, while a_k is the coefficient

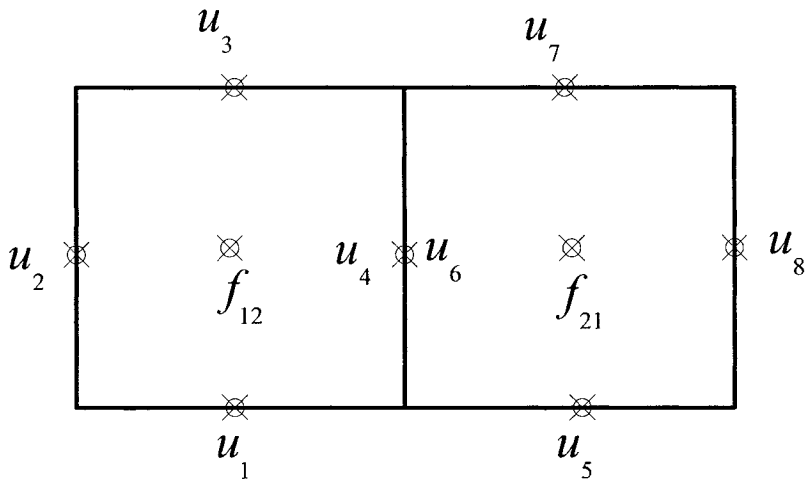


Figure 5.5: A 2 by 1 mesh in a 2D domain

of u_{12} $(= u_4)$, and $a_l^{'}$, is the coefficient of u_{21} $(= u_6)$. Here, we use the superscript "′" to denote the second element. As $a_k = A_{22}^1$ and $a_l^{'} = A_{22}^2$ are diagonal terms, using Equation (5.38), we can write

$$a_k = -\sum_{i=1, i\neq k}^{4} a_i,$$

and

$$a_l^{'} = -\sum_{j=1, j\neq l}^{4} a_j^{'}.$$

Thus, the sum of $a_k + a_l^{'}$ which is the coefficient of u_{12} $(= u_{21})$ is equal to *the negative value of the sum of other terms*, *i.e.*,

$$a_k + a_l^{'} = -\sum_{i=1, i\neq k}^{4} a_i - \sum_{j=1, j\neq l}^{4} a_j^{'}. \tag{5.40}$$

In the current example we have

$$4h^{-1} = -(-h^{-1} - h^{-1} - 0 - h^{-1} - h^{-1} - 0). \tag{5.41}$$

The same procedure can be used for four elements, corresponding to a 2 by 2 mesh with one collocation point, or any general **M** by **N** mesh. In any such case, for the matching point between any two adjacent elements, we can write Equation (5.34), simply by adding $v_{12}+v_{21}$ and find an equation which only depends on "u" coefficients. The coefficient of u_{12}, which equals $a_k + a_l^{'}$, is equal to the negative value of the sum of other off-diagonal coefficients, *i.e.*, Equation (5.40) is valid. As a result, if we collect

this equation for every interior (matching) points of the mesh, then we have a system of equations only in terms of "u"s. For a grand total of M interior matching points of all elements and N total boundary and interior matching points, we obtain a system of the following form

$$A_{M*N}U_{N*1} = F_{M*1}. \tag{5.42}$$

This system is not square. For solving System (5.42), we need to add the boundary condition (Equations (5.36)). For each boundary point i, $M < i \leq N$, we add a line to the above system such that $a_{ij} = 0$ if $i \neq j$, $a_{ii} = 1$ and $f_i = 0$. As a result, we have a $N * N$ system that satisfies

$$\mid a_{ii} \mid \geq \sum_{j=1, j\neq i}^{N} \mid a_{ij} \mid . \tag{5.43}$$

Inequality is for the case of the boundary condition equations. As one can see, this system is diagonally dominant (*cf.* [149]). The system (5.42) can also be written as

$$A^1_{M*M}U^1_{M*1} + A^2_{M*(N-M)}U^2_{(N-M)*1} = F_{M*1},$$

or

$$A^1_{M*M}U^1_{M*1} = F_{M*1} - A^2_{M*(N-M)}U^2_{(N-M)*1} = F^1, \tag{5.44}$$

where $U^2_{(n-m)*1}$ are the known boundary values. The matrix A^1_{m*m} of the above system still satisfies the following condition

$$\mid a_{ii} \mid \geq \sum_{j=1, j\neq i}^{M} \mid a_{ij} \mid, \tag{5.45}$$

because, for $1 \leq i \leq M$, if we eliminate some a_{ij}, $i \neq j$, which are the boundary value coefficients, from Equation (5.43) the inequality is still valid. This system is also irreducible, because apart from the case of a 2 by 1 mesh, for which we do not have off-diagonal terms (the matrix A^1_{M*M} is 1 by 1, and has only one coefficient), each row of the matrix A^1_{M*M} has at least one off-diagonal term. As a result, the graph defined by the matrix A^1_{M*M} is strongly connected. Therefore, it is irreducible and diagonally dominant, and we can conclude that System (5.44) is nonsingular [149].

The case of one collocation point per element (Equation (5.41)) gives an equation which is the same as the classical five-point approximation finite difference schema. The coefficient of the common matching point u_{12} is equal to the negative value of the

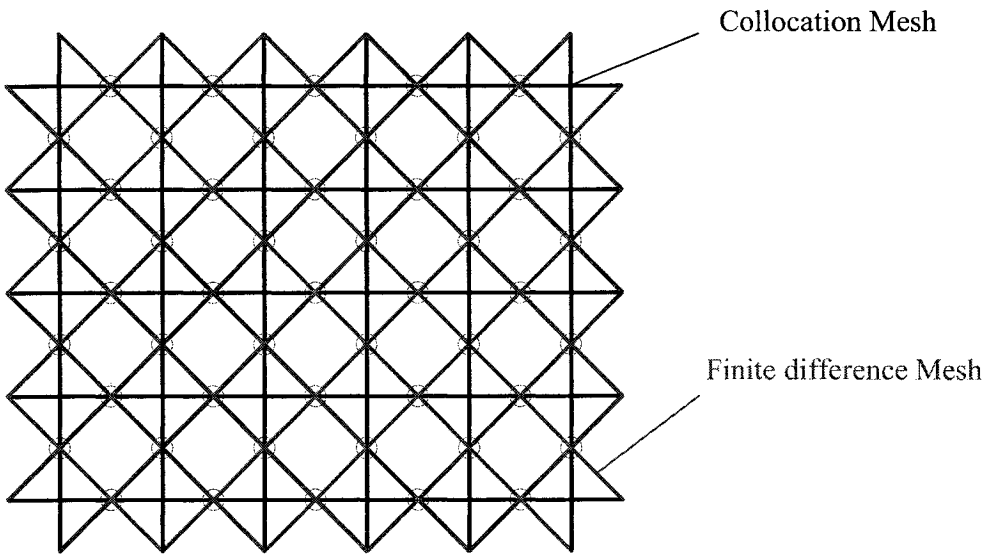


Figure 5.6: The relation between the collocation and the finite difference mesh

sum of coefficients of four nearest matching points. At first look, the positions of the matching points seem different from the finite difference mesh points, but if we turn the finite difference mesh 45 degree with respect to the collocation mesh, then we have identical positions (Figure 5.6). The differences between these two approaches are:

- First, the positions where we evaluate the function f (Equation (5.35)) are not the same. In the finite difference scheme f is evaluated at the center point, but in the collocation method f is evaluated at the center of each of the two adjacent elements (Figure 5.7), and we have

$$g_{12}^1 + g_{21}^2 = (h/4)f_{12} + (h/4)f_{21} = h(f_{12} + f_{21})/4.$$

- Second, the mesh sizes are different. As one can see, the size of finite difference mesh is $\frac{\sqrt{2}}{2}$ times the size of collocation method mesh (h).

Now consider a 1 by 2 mesh with 4 collocation points (Figure 5.8). We place the collocation points at the Gauss positions . In this case, for each element, the matrix

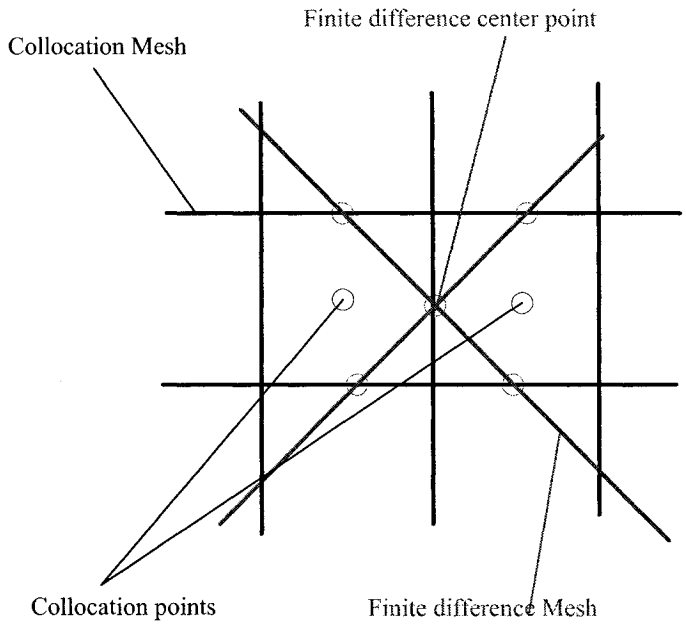


Figure 5.7: Evaluation points of the collocation and the finite difference methods

A in Equation (5.5) is given by

$$\begin{bmatrix}
4\frac{1}{h} & -\frac{3}{4}\frac{1}{h} & -\frac{1}{4}\frac{1}{h} & 0 & c_1 & 0 & 0 & c_2 \\
-\frac{3}{4}\frac{1}{h} & 4\frac{1}{h} & 0 & -\frac{1}{4}\frac{1}{h} & 0 & c_2 & c_1 & 0 \\
-\frac{1}{4}\frac{1}{h} & 0 & 4\frac{1}{h} & -\frac{3}{4}\frac{1}{h} & 0 & c_1 & c_2 & 0 \\
0 & -\frac{1}{4}\frac{1}{h} & -\frac{3}{4}\frac{1}{h} & 4\frac{1}{h} & c_2 & 0 & 0 & c_1 \\
c_1 & 0 & 0 & c_2 & 4\frac{1}{h} & -\frac{3}{4}\frac{1}{h} & -\frac{1}{4}\frac{1}{h} & 0 \\
0 & c_2 & c_1 & 0 & -\frac{3}{4}\frac{1}{h} & 4\frac{1}{h} & 0 & -\frac{1}{4}\frac{1}{h} \\
0 & c_1 & c_2 & 0 & -\frac{1}{4}\frac{1}{h} & 0 & 4\frac{1}{h} & -\frac{3}{4}\frac{1}{h} \\
c_2 & 0 & 0 & c_1 & 0 & -\frac{1}{4}\frac{1}{h} & -\frac{3}{4}\frac{1}{h} & 4\frac{1}{h}
\end{bmatrix},$$

with $c_1 = -\frac{3}{4}\frac{2+\sqrt{3}}{h}$ and $c_2 = \frac{3}{4}\frac{-2+\sqrt{3}}{h}$. As for the case of one collocation point, the matrix A is diagonally dominant. For any coefficient a_{ij}, $i \neq j$, we have $a_{ij} \leq 0$ and $a_{ii} > 0$. The sum of off-diagonal terms is exactly equal to the negative value of the diagonal term and, Equation (5.38) is valid (with $n = 8$). As before,

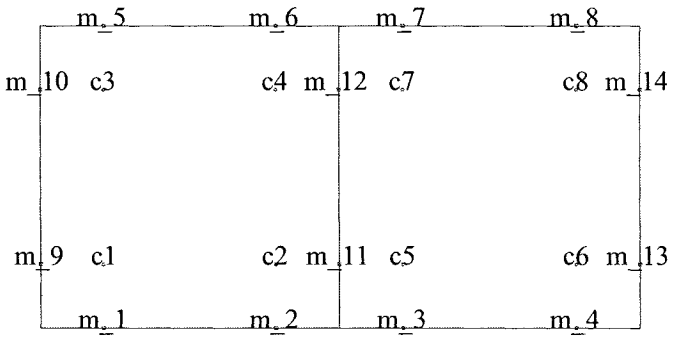


Figure 5.8: A 1 by 2 mesh with 4 collocation points per element

we can collect Equation (5.34) for every common matching points between adjacent elements. Adding the equations corresponding to the common matching points, one obtains an equation in terms of the "u"s only. Here, the common side between two adjacent elements has 2 matching points. Denoting, as before, a_k as a matching point coefficient of first element, and $a_l^{'}$ as the corresponding coefficient of the second element, we have

$$a_k + a_l^{'} = -\sum_{i=1,i\neq k}^{8} a_i - \sum_{j=1,j\neq l}^{8} a_j^{'}. \tag{5.46}$$

Writing Equation (5.34) for two adjacent elements, using the indices shown in Figure 5.8, we have

$$\begin{aligned}
&\frac{-1}{4h}((6+3\sqrt{3})u_2 + (6+3\sqrt{3})u_3 + (6-3\sqrt{3})u_5 + (6-3\sqrt{3})u_8 + \\
&u_9 - 32\,u_{11} + 6\,u_{12} + u_{13}) - \frac{3+\sqrt{3}}{96}((4\sqrt{3}-7)f_1 - (2+\sqrt{3})f_2 - f_4 \\
&+(\sqrt{3}-2)f_3 - (2+\sqrt{3})f_5 + (4\sqrt{3}-7)f_6 - f_7 + (\sqrt{3}-2)f_8)h = 0,
\end{aligned} \tag{5.47}$$

and

$$\begin{aligned}
&\frac{-1}{4h}((6-3\sqrt{3})u_1 + (6-3\sqrt{3})u_4 + (6+3\sqrt{3})u_6 + (6+3\sqrt{3})u_7 + u_{10} \\
&+6\,u_{11} - 32\,u_{12} + u_{14}) - \frac{9+5\sqrt{3}}{96}((4\sqrt{3}-7)f_1 + (\sqrt{3}-2)f_2 - f_4 - f_7 \\
&+(15\sqrt{3}-26)f_3 + (\sqrt{3}-2)f_5 + (4\sqrt{3}-7)f_6 + (15\sqrt{3}-26)f_8)h = 0.
\end{aligned} \tag{5.48}$$

The coefficients of u_i, $i = 1, \cdots, 14$ for the first matching point are

$$\begin{aligned}
&\Big[0, -\frac{3}{4}\frac{2+\sqrt{3}}{h}, -\frac{3}{4}\frac{2+\sqrt{3}}{h}, 0, \frac{3}{4}\frac{-2+\sqrt{3}}{h}, 0, \\
&0, \frac{3}{4}\frac{-2+\sqrt{3}}{h}, -\frac{1}{4}\frac{1}{h}, 0, 8\frac{1}{h}, -\frac{3}{2}\frac{1}{h}, -\frac{1}{4}\frac{1}{h}, 0\Big],
\end{aligned}$$

and for the second matching point they are given by

$$\left[\frac{3}{4}\frac{-2+\sqrt{3}}{h}, 0, 0, \frac{3}{4}\frac{-2+\sqrt{3}}{h}, 0, -\frac{3}{4}\frac{2+\sqrt{3}}{h}, -\frac{3}{4}\frac{2+\sqrt{3}}{h}, 0, 0, -\frac{1}{4}\frac{1}{h}, -\frac{3}{2}\frac{1}{h}, 8\frac{1}{h}, 0, -\frac{1}{4}\frac{1}{h}\right].$$

The common matching point coefficient in each of above lines is $8\,h^{-1}$, which is equal to the negative value of the sum of the remaining terms. The same procedure can be applied to any **M** by **N** mesh with four collocation points per element. For any matching point between any two adjacent elements, Equation (5.46) is valid. If we write Equation (5.34) for all interior matching points of the mesh, we construct a system in terms of the "u"s only. For a total of M interior matching points of all elements and N boundary and interior matching points, we can write a system of the form of Equation (5.42) or Equation (5.44). To complete this system, we need to add the boundary condition equations. As for the case of one collocation point per element, the condition (5.43) or (5.45) is valid. Each line of the matrix A^1_{M*M} has at least one off-diagonal term. As a result, the system is irreducible, diagonally dominant, from which it follows that this system is nonsingular and has a unique solution [149].

Unfortunately, the above diagonally dominant conclusion is not valid for any number or any position of collocation points, because, in the general case, the conditions "$a_{ij} \leq 0,\ i \neq j$ and $a_{ii} > 0$" may not be satisfied. In the case of four collocation points per element, one can use a parameter α to define the following positions of the collocation points

$$\begin{aligned} p_{c1} &= (\quad \alpha h, \quad \alpha h), \\ p_{c2} &= (\quad \alpha h, (1-\alpha)h), \\ p_{c3} &= ((1-\alpha)h, \quad \alpha h), \\ p_{c4} &= ((1-\alpha)h, (1-\alpha)h), \end{aligned}$$

and the matching points

$$p_{m1} = (\ \alpha h, \quad 0),$$

$$
\begin{aligned}
p_{m2} &= ((1-\alpha)h, 0), \\
p_{m3} &= (\ \ h, \ \ \alpha h), \\
p_{m4} &= (h, (1-\alpha)h), \\
p_{m5} &= ((1-\alpha)h, h), \\
p_{m6} &= (\alpha h, \ \ \ \ h), \\
p_{m7} &= (0, (1-\alpha)h), \\
p_{m8} &= (\ \ 0, \ \ \alpha h).
\end{aligned}
$$

The range of α which satisfies the above diagonally dominant condition is from $\alpha = 0.14675h$ until $\alpha = 0.21925h$. For example, if we equally space the matching points ($\alpha = 0.25$), instead of Gauss points ($\alpha = 0.211324865$), then the matrix A, in Equation (5.49), is not diagonally dominant.

$$
A = \frac{1}{h}\begin{bmatrix}
11/3 & -2/3 & -1/3 & 0 & -5/2 & 1/6 & -1/6 & -1/6 \\
-2/3 & 11/3 & 0 & -1/3 & -1/6 & -1/6 & -5/2 & 1/6 \\
-1/3 & 0 & 11/3 & -2/3 & 1/6 & -5/2 & -1/6 & -1/6 \\
0 & -1/3 & -2/3 & 11/3 & -1/6 & -1/6 & 1/6 & -5/2 \\
-5/2 & 1/6 & -1/6 & -1/6 & 11/3 & -2/3 & -1/3 & 0 \\
-1/6 & -1/6 & -5/2 & 1/6 & -2/3 & 11/3 & 0 & -1/3 \\
1/6 & -5/2 & -1/6 & -1/6 & -1/3 & 0 & 11/3 & -2/3 \\
-1/6 & -1/6 & 1/6 & -5/2 & 0 & -1/3 & -2/3 & 11/3
\end{bmatrix}. \qquad (5.49)
$$

Although, the sum of off-diagonal terms is exactly equal to the negative value of the diagonal term, we do not have $a_{ij} \leq 0$ for any coefficient a_{ij}, $i \neq j$. For example, there is a positive off-diagonal term $1/6\,h^{-1}$. For a 3 by 3 collocation scheme we have not found a placement scheme that satisfies the diagonally dominant condition. As a result, we can not use the same proof. This does not mean that the system is singular, as, for example, the matrix A in Equation (5.49) is nonsingular. Another type of proof will be required for these situations.

For a 2 by 2 collocation scheme with equal spacing, Equation (5.34) can still be used, and Equation (5.46) is also valid. The coefficients of u_i, $i = 1, \cdots, 14$, (see Figure 5.8) for the first matching point are

$$[\frac{1}{6h}, -\frac{5}{2h}, -\frac{5}{2h}, \frac{1}{6h}, -\frac{1}{6h}, -\frac{1}{6h}, -\frac{1}{6h}, -\frac{1}{6h}, -\frac{1}{3h}, 0, \frac{22}{3h}, -\frac{4}{3h}, -\frac{1}{3h}, 0],$$

and for the second matching point they are

$$[-\frac{1}{6h}, -\frac{1}{6h}, -\frac{1}{6h}, -\frac{1}{6h}, \frac{1}{6h}, -\frac{5}{2h}, -\frac{5}{2h}, \frac{1}{6h}, 0, -\frac{1}{3h}, -\frac{4}{3h}, \frac{22}{3h}, 0, -\frac{1}{3h}].$$

The term $22/3h$ in each of the above lists is the coefficient of the common matching point. Note that the sum of the other terms also equals $-22/3h$, but we have a positive a_{ij}, $i \neq j$, term $1/6h$.

Reconsider Equations (5.47) and (5.48), for the case of 4 collocation points per element, where the collocation points are placed at Gauss positions. We can apply Taylor series expansion of each term on the left hand side of these two equations to find the error term of our approximation. Expansions are done with respect to a common point on the line separating adjacent elements; for example, we can use the upper end point of this line ($x_0 = h$ and $y_0 = h$). As the right hand sides of these equations are equal to zero, the sum of left hand side terms give us the error terms. These errors, for the points 11 and 12 of Figure 5.8, are found to be

$$\begin{aligned} \epsilon_1 &= \frac{1}{8640} h^3 (h(\sqrt{3} - 30)\frac{\partial^5}{\partial y^5} w(h,h) + h(30 + 15\sqrt{3})\frac{\partial^5}{\partial x^4 \partial y} w(h,h) \quad (5.50) \\ &\quad +60\frac{\partial^4}{\partial y^4} w(h,h) - 60\frac{\partial^4}{\partial x^4} w(h,h)) + \mathcal{O}(h^5), \end{aligned}$$

and

$$\begin{aligned} \epsilon_2 &= \frac{-1}{8640} h^3 (h(30 + \sqrt{3})\frac{\partial^5}{\partial y^5} w(h,h) + h(15\sqrt{3} - 30)\frac{\partial^5}{\partial x^4 \partial y} w(h,h) \quad (5.51) \\ &\quad -60\frac{\partial^4}{\partial y^4} w(h,h) + 60\frac{\partial^4}{\partial x^4} w(h,h)) + \mathcal{O}(h^5). \end{aligned}$$

Here, we use w to present the exact value and u to present the approximate solution. If two adjacent elements are above each other, as in Figure 5.9, these errors for the points 3 and 4 are

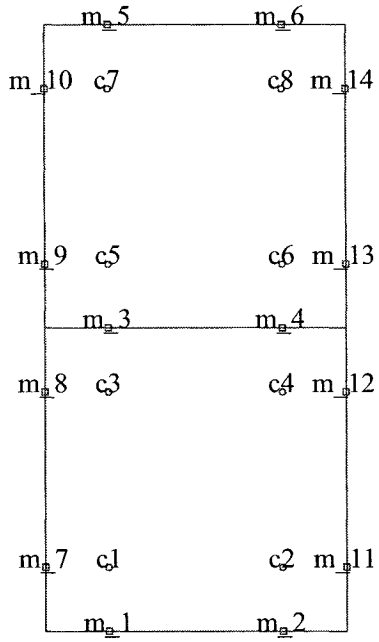


Figure 5.9: A 2 by 1 mesh with 4 collocation points per element

$$\epsilon_1 = \frac{1}{8640}\, h^3(h(\sqrt{3}-30)\frac{\partial^5}{\partial x^5}w(h,h) + h(30+15\sqrt{3})\frac{\partial^5}{\partial y^4 \partial x}w(h,h) \quad (5.52)$$
$$+60\,\frac{\partial^4}{\partial x^4}w(h,h) - 60\,\frac{\partial^4}{\partial y^4}w(h,h)) + \mathcal{O}(h^5),$$

and

$$\epsilon_2 = \frac{-1}{8640}\, h^3(h(30+\sqrt{3})\frac{\partial^5}{\partial x^5}w(h,h) + h(15\sqrt{3}-30)\frac{\partial^5}{\partial y^4 \partial x}w(h,h) \quad (5.53)$$
$$-60\,\frac{\partial^4}{\partial x^4}w(h,h) + 60\,\frac{\partial^4}{\partial y^4}w(h,h)) + \mathcal{O}(h^5).$$

Therefore, in either case, if we take

$$\max(|\frac{\partial^5 w}{\partial x^5}|, |\frac{\partial^5 w}{\partial y^5}|, |\frac{\partial^5 w}{\partial y^4 \partial x}|, |\frac{\partial^5 w}{\partial x^4 \partial y}|\ ) \le \mathcal{M}, \qquad (x,y) \in \partial\Omega, \quad (5.54)$$

then the maximum error satisfies

$$\epsilon \le \frac{1}{144}\, h^3 |\frac{\partial^4 w}{\partial x^4} - \frac{\partial^4 w}{\partial y^4}| + \frac{1}{4320}\left(7\sqrt{3}+30\right) h^4 \mathcal{M} + \mathcal{O}(h^5), \qquad (x,y) \in \partial\Omega. \quad (5.55)$$

If we define

$$\bar{\epsilon} = (\epsilon_1, \epsilon_2, \cdots, \epsilon_N)^T,$$

$$W = (w_1, w_2, \cdots, w_N)^T,$$

$$U = (u_1, u_2, \cdots, u_N)^T,$$

$$F = (f_1, f_2, \cdots, f_N)^T,$$

where N is the number of matching and boundary points over the whole regain, then we can write error terms for all matching points, and present them in the matrix form

$$AW = F - \overline{\epsilon}. \tag{5.56}$$

From Equation (5.42), after adding the boundary conditions, we have

$$AU = F, \tag{5.57}$$

and thus

$$A(W - U) = -\overline{\epsilon}. \tag{5.58}$$

From the previous discussion we know that A is nonsingular. Thus we can solve the above system for the error terms $\overline{\omega} = W - U$. Here $\omega_i = w_i - u_i$ and $\overline{\omega} = (\omega_1, \omega_2, \cdots, \omega_N)^T$. As $w = u = 0$ on $\partial\Omega$, we have $\omega_i = 0$ for $x_i \in \partial\Omega$. As a result we can write

$$|w_i - u_i| = |\omega_i| \leq \varphi_i, \tag{5.59}$$

where φ_i, $i = 1, \cdots, N$, are quantities determined by the equation

$$A\overline{\varphi} = -S,$$

where $s_i \geq |\epsilon_i|$. Here $\overline{\varphi} = (\varphi_1, \varphi_2, \cdots, \varphi_N)^T$ and $S = (s_1, s_2, \cdots, s_N)^T$. φ_i can be obtained by solving the equation $A\overline{\hat{\varphi}} = -1$, and setting $\overline{\varphi} = \epsilon * \overline{\hat{\varphi}}$. The value of ϵ is defined by Equation (5.55).

If we can find a set $\hat{\varphi}_i$, $\overline{\hat{\varphi}} = (\hat{\varphi}_1, \hat{\varphi}_2, \cdots, \hat{\varphi_N})^T$ such that

$$-A\overline{\hat{\varphi}} = \overline{\alpha},$$

where $\overline{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_N)^T$, so that all $\alpha_i > 0$, for all i, then we do not need to solve the system $A\overline{\hat{\varphi}} = -1$ (*cf.* [32]), because we can write

$$|\omega_i| \leq \varphi_i = \epsilon\hat{\varphi}_i / \min_{1 \leq i \leq N}(\alpha_i). \tag{5.60}$$

For our problem, Equation (5.35), if we take

$$\hat{\varphi}_i = (x_i + h)^2,$$

where x_i is the local coordinate of the matching point i, *i.e.*, in Figure 5.8 we choose the left bottom corner as the center of the local coordinate system, then all α_i except at domain boundary points, will be equal to h. As an example, α_i corresponding to Equation (5.47) is equal to

$$-\frac{-1}{4h}((6+3\sqrt{3})(u_2+u_3)+(6-3\sqrt{3})(u_5+u_8)+u_9-32\,u_{11}+6\,u_{12}+u_{13}),$$

or

$$\begin{aligned}\frac{1}{4h}((6+3\sqrt{3})((h+\frac{3+\sqrt{3}}{6}h)^2+(2h+\frac{3-\sqrt{3}}{6}h)^2) \quad &+\\ (6-3\sqrt{3})((h+\frac{3-\sqrt{3}}{6}h)^2+(2h+\frac{3+\sqrt{3}}{6}h)^2) \quad &+\\ (h+0)^2-32\,(h+h)^2+6\,(h+h)^2+(h+2h)^2) = \quad & h.\end{aligned}$$

For each row of the matrix A corresponding to a boundary point x_i, as mentioned before, all a_{ij}, with $i \neq j$, are equal to 0 except a_{ii} which is equal to 1. Therefore, the corresponding α_i is equal to $\hat{\varphi}_i = (x_i+h)^2$. At boundary points $\alpha_i = (x_i+h)^2$ is always positive. The minimum value of α_i is equal to h^2 (when $x_i = 0$). The maximum value of $\hat{\varphi}_i$ is equal to $(h+2h)^2 = 9h^2$ (when $x_i = 2h$). Therefore we can write

$$|\omega_i| \leq \varphi_i \leq \frac{\epsilon(3h)^2}{h^2} = 9\epsilon. \tag{5.61}$$

Using Equations (5.55) and (5.59), we have

$$|w_i - u_i| \leq 9(\frac{h^3}{144}|\frac{\partial^4 w}{\partial x^4} - \frac{\partial^4 w}{\partial y^4}| + \frac{7\sqrt{3}+30}{4320}\,h^4\mathcal{M} + \mathcal{O}(h^5)), \qquad (x,y) \in \partial\Omega, \tag{5.62}$$

or

$$|w_i - u_i| \leq \frac{h^3}{16}|\frac{\partial^4 w}{\partial x^4} - \frac{\partial^4 w}{\partial y^4}| + \frac{7\sqrt{3}+30}{480}\,h^4\mathcal{M} + \mathcal{O}(h^5), \qquad (x,y) \in \partial\Omega. \tag{5.63}$$

We see that as $h \to 0$ the solution converges to the exact solution, with error of order $\mathcal{O}(h^3)$. Moreover, if we have

$$\frac{\partial^4 w}{\partial x^4} - \frac{\partial^4 w}{\partial y^4} = 0,$$

or its equivalent

$$\frac{\partial^2 f}{\partial x^2} - \frac{\partial^2 f}{\partial y^2} = 0,$$

which is satisfied for symmetric f, where f is the right hand of Equation (5.35), then we have convergence order of $\mathcal{O}(h^4)$. Numerical results in Chapter 11, Sections 11.1 and 11.2 confirm this order.

Chapter 6

The Collocation Method for Nonlinear Elliptic PDE BVP

6.1 Introduction

As mentioned in Section 4.1 for ODE BVPs, the finite difference approach cannot be used for nonlinear problems, as a linear relation of the form Equation (5.3) does not exist between the variable u and its derivative v at boundary points of an element. However, the collocation formulation described in Section 5.4 for the linear PDEs can be used for nonlinear PDEs. Although, we are considering nonlinear PDEs, the main features of the procedure stay the same as for linear PDEs.

6.2 Collocation with discontinuous piecewise polynomials

Consider the second order nonlinear PDE

$$Nu \equiv \Delta u + f(x, u, \nabla u) = 0, \qquad x \in \Omega \subset R^2, \tag{6.1}$$

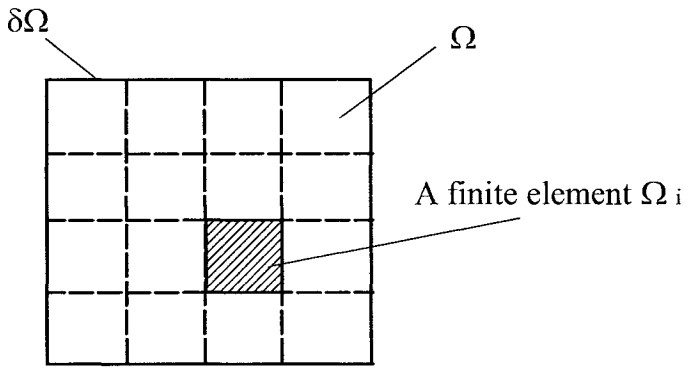


Figure 6.1: A 2D region Ω, its recursive subdivision, and a finite element mesh

where Δ is the Laplace operator, ∇u is the gradient of u and $f(x, u, \nabla u), u(x) \in R$. As the boundary condition we take

$$u(x) = 0, \qquad x \in \delta\Omega. \tag{6.2}$$

As in the previous chapter, the implementation is for a simple boundary condition, but we can use more general boundary conditions, as presented in Chapter 11. In the mesh generation step, as in Chapter 5, the domain Ω is recursively subdivided. A polynomial $p(x) \in P_{n+m}$ is associated to each finite element, where m is the number of collocation points, and n is the number of matching points for an element. The polynomial $p(x)$, first must satisfy the collocation equations

$$Np(z_k) = 0, \qquad k = 1, \cdots, m.$$

Second, for any two adjacent finite elements, and for any point x_i at the common boundary between these two elements, the value and normal derivative of the neighboring polynomials must match.

For a finite element the local polynomial has the form $p(x) = \sum_{i=1}^{n+m} c_i \phi_i(x)$, where $Span\{\phi_1, \cdots, \phi_{n+m}\} = P_{n+m}$. The collocation equations are

$$N(\sum_{i=1}^{n+m} c_i \phi_i(z_k)) = 0, \qquad k = 1, \cdots, m. \tag{6.3}$$

To satisfy the continuity requirements between adjacent elements, unique variables u_i and v_i are associated to each matching point x_i such that $p(x_i) = u_i$ and $\nabla p(x_i)^T \eta_i = v_i$. We assume that for any given x_i the outward normal η_i has a unique orientation.

Using the notation of the previous chapter, we have

$$\begin{aligned} u - \Phi^T c &= 0, \\ v - R_\Phi^T c &= 0. \end{aligned} \tag{6.4}$$

where

$$\Phi = \begin{pmatrix} \phi_1(x_1) & \cdots & \phi_1(x_n) \\ \vdots & & \vdots \\ \phi_{n+m}(x_1) & \cdots & \phi_{n+m}(x_n) \end{pmatrix}, \tag{6.5}$$

$$R_\Phi = \begin{pmatrix} \nabla\phi_1(x_1)^T\eta_1 & \cdots & \nabla\phi_1(x_n)^T\eta_n \\ \vdots & & \vdots \\ \nabla\phi_{n+m}(x_1)^T\eta_1 & \cdots & \nabla\phi_{n+m}(x_n)^T\eta_n \end{pmatrix}. \tag{6.6}$$

6.3 Newton's method

Equations (6.3) and (6.4) of all elements together with the discrete boundary conditions constitute the discretization. The unknowns are $c \in R^{n+m}$ for each finite element, and the u_i and v_i associated with the points x_i on the inter-element boundaries. As in Section 4.4, to solve (6.3),(6.4) for u, v, and c, we use Newton's method. Omitting iteration indices, we have

$$L_\Phi^T \, \delta c = -r_N, \tag{6.7}$$

$$\begin{aligned} &\text{a)} \quad \delta u - \Phi^T \delta c = -r_u, \\ &\text{b)} \quad \delta v - R_\Phi^T \delta c = -r_v, \end{aligned} \tag{6.8}$$

which

$$L_\Phi^T = \begin{pmatrix} L[p(z_1)]\phi_1(z_1) & \cdots & L[p(z_1)]\phi_{n+m}(z_1) \\ \vdots & & \vdots \\ L[p(z_m)]\phi_1(z_m) & \cdots & L[p(z_m)]\phi_{n+m}(z_m) \end{pmatrix}, \tag{6.9}$$

where L is the linearization of N, *i.e.*, $L[p]\phi(z)$ is the linearization of N about p acting on ϕ at z, more precisely,

$$L[p]\phi(z) = \Delta\phi(z) + D_2 f(z, p(z), \nabla p(z))\phi(z) + D_3 f(z, p(z), \nabla p(z))^T \nabla\phi(z).$$

Further we define

$$\delta c = \begin{pmatrix} \delta c_1 \\ \vdots \\ \delta c_{n+m} \end{pmatrix}, \quad r_N = \begin{pmatrix} Np(z_1) \\ \vdots \\ Np(z_m) \end{pmatrix}, \quad \delta u = \begin{pmatrix} \delta u_1 \\ \vdots \\ \delta u_n \end{pmatrix}, \quad \delta v = \begin{pmatrix} \delta v_1 \\ \vdots \\ \delta v_n \end{pmatrix}, \tag{6.10}$$

and

$$r_u = u - \Phi^T c, \qquad r_v = v - R_\Phi^T c.$$

Equations (6.7) and (6.8a) can be written

$$\begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix} \delta c = \begin{pmatrix} \delta u + r_u \\ -r_N \end{pmatrix}. \tag{6.11}$$

Using (6.11) to eliminate δc in (6.8b) one obtains

$$\delta v = R_\Phi^T \begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix}^{-1} \begin{pmatrix} \delta u + r_u \\ -r_N \end{pmatrix} - r_v.$$

Define A and B precisely as in the previous chapter,

$$A = \begin{pmatrix} \alpha_{11} & \cdots & \alpha_{1n} \\ \vdots & & \vdots \\ \alpha_{n1} & \cdots & \alpha_{nn} \end{pmatrix}, \qquad B = \begin{pmatrix} \beta_{11} & \cdots & \beta_{1m} \\ \vdots & & \vdots \\ \beta_{n1} & \cdots & \beta_{nm} \end{pmatrix},$$

which are the solutions of the following linear system

$$(\Phi \mid L_\Phi) \begin{pmatrix} A^T \\ B^T \end{pmatrix} = R_\Phi. \tag{6.12}$$

Then, the above expression for δv can be rewritten as

$$\delta v = (A \quad B) \begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix} \begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix}^{-1} \begin{pmatrix} \delta u + r_u \\ -r_N \end{pmatrix} - r_v,$$

that is,

$$\delta v = A\delta u - Br_N - r_v + Ar_u. \tag{6.13}$$

This equation has the same form of Equation (5.5) of the previous chapter.

6.4 Nested dissection

Equation (6.13) is similar to Equations (5.5), (4.12) and (3.8). Therefore, we can use the nested dissection method described in Sections 4.4.1 and 5.7 to eliminate the unknowns δu and δv on boundaries separating adjacent regions. In this procedure, all the interior unknowns will be eliminated. One is left with one equation for each x_i on $\delta\Omega$. Afterward, the boundary conditions are used to determine the values of δu and δv on $\delta\Omega$. A recursive back-substitution gives the values of δu and δv on each interior matching point. As this procedure has already been described in the previous

chapter for linear PDEs, we mention only briefly the equivalent steps and equations for nonlinear PDEs.

Equation (6.13) is rewritten as follow:

$$D\delta v = A\delta u + g, \tag{6.14}$$

where

$$g = -Br_N - r_v + Ar_u.$$

The δu and δv are vectors here. Matrix D is an identity matrix at the element level, but it is not necessary an identity matrix at the interior nodes of the binary tree during the execution of the algorithm. Equation (6.14) can be written in the split form for regions Ω_1 and Ω_2. (refer to Figure 5.4)

$$\begin{pmatrix} D^1_{11} & D^1_{12} \\ D^1_{21} & D^1_{22} \end{pmatrix} \begin{pmatrix} \delta v_1 \\ \delta v_{12} \end{pmatrix} = \begin{pmatrix} A^1_{11} & A^1_{12} \\ A^1_{21} & A^1_{22} \end{pmatrix} \begin{pmatrix} \delta u_1 \\ \delta u_{12} \end{pmatrix} + \begin{pmatrix} g^1_1 \\ g^1_{12} \end{pmatrix}, \tag{6.15}$$

$$\begin{pmatrix} D^2_{11} & D^2_{12} \\ D^2_{21} & D^2_{22} \end{pmatrix} \begin{pmatrix} \delta v_2 \\ \delta v_{21} \end{pmatrix} = \begin{pmatrix} A^2_{11} & A^2_{12} \\ A^2_{21} & A^2_{22} \end{pmatrix} \begin{pmatrix} \delta u_2 \\ \delta u_{21} \end{pmatrix} + \begin{pmatrix} g^2_2 \\ g^2_{21} \end{pmatrix}, \tag{6.16}$$

where

$$\begin{aligned} g^1_1 &= -B^1_1 r^1_N - r_{v1} + A^1_{11} r_{u1} + A^1_{12} r_{u12}, \\ g^1_{12} &= -B^1_2 r^1_N - r_{v12} + A^1_{21} r_{u1} + A^1_{22} r_{u12}, \\ g^2_2 &= -B^2_1 r^2_N - r_{v2} + A^2_{11} r_{u2} + A^2_{12} r_{u21}, \\ g^2_{21} &= -B^2_2 r^2_N - r_{v21} + A^2_{21} r_{u2} + A^2_{22} r_{u21}. \end{aligned}$$

The superscript represents the region number. From the continuity relations, the values and the derivatives of the neighboring polynomials must match, therefore we can write

$$\begin{aligned} \delta u_{12} &= \delta u_{21}, \\ \delta v_{12} &= -\delta v_{21}. \end{aligned} \tag{6.17}$$

Because δv is an outward normal to the boundary at the matching point, the direction of δv_{12} is opposite to the direction of δv_{21}. As result, we have $\delta v_{12} = -\delta v_{21}$. The

above system can also be written as:

$$\begin{pmatrix} D_{22}^1 & -A_{22}^1 & D_{21}^1 & -A_{21}^1 & 0 & 0 \\ -D_{22}^2 & -A_{22}^2 & 0 & 0 & D_{21}^2 & -A_{21}^2 \\ D_{12}^1 & -A_{12}^1 & D_{11}^1 & -A_{11}^1 & 0 & 0 \\ -D_{12}^2 & -A_{12}^2 & 0 & 0 & D_{11}^2 & -A_{11}^2 \end{pmatrix} \begin{pmatrix} \delta v_{12} \\ \delta u_{12} \\ \delta v_1 \\ \delta u_1 \\ \delta v_2 \\ \delta u_2 \end{pmatrix} = \begin{pmatrix} g_{12}^1 \\ g_{21}^2 \\ g_1^1 \\ g_2^2 \end{pmatrix}. \tag{6.18}$$

Now, we use a full pivoting strategy, as described in Section 5.7 and Section 4.4.1, to obtain the following system.

$$\begin{pmatrix} I & 0 & \hat{D}_{11} & \hat{A}_{11} & \hat{D}_{12} & \hat{A}_{12} \\ 0 & I & \hat{D}_{21} & \hat{A}_{21} & \hat{D}_{22} & \hat{A}_{22} \\ 0 & 0 & D_{11} & C_{11} & D_{12} & C_{12} \\ 0 & 0 & D_{21} & C_{21} & D_{22} & C_{22} \end{pmatrix} \begin{pmatrix} \delta v_{12} \\ \delta u_{12} \\ \delta v_1 \\ \delta u_1 \\ \delta v_2 \\ \delta u_2 \end{pmatrix} = \begin{pmatrix} \hat{g}_{12} \\ \hat{g}_{21} \\ \hat{g}_1 \\ \hat{g}_2 \end{pmatrix}. \tag{6.19}$$

From Equation (6.19), we can write

$$\delta u_{12} = \hat{g}_{21} - (\hat{D}_{21}\delta v_1 + \hat{A}_{21}\delta u_1 + \hat{D}_{22}\delta v_2 + \hat{A}_{22}\delta u_2), \tag{6.20}$$

$$\delta v_{12} = \hat{g}_{12} - (\hat{D}_{11}\delta v_1 + \hat{A}_{11}\delta u_1 + \hat{D}_{12}\delta v_2 + \hat{A}_{12}\delta u_2), \tag{6.21}$$

and

$$\begin{pmatrix} D_{11} & C_{11} & D_{12} & C_{12} \\ D_{21} & C_{21} & D_{22} & C_{22} \end{pmatrix} \begin{pmatrix} \delta v_1 \\ \delta u_1 \\ \delta v_2 \\ \delta u_2 \end{pmatrix} = \begin{pmatrix} \hat{g}_1 \\ \hat{g}_2 \end{pmatrix}. \tag{6.22}$$

Equation (6.22) represents the discrete equations for the enlarged region, *i.e.*, for the union of Ω_1 and Ω_2, after the elimination of the common boundary unknowns δu_{12} and δv_{12}. These new equations are again of the form Equation (6.14).

6.5 Special basis functions

The special basis functions described in Section 6.5 can also be used to solve the system (6.12) efficiently. If the discretization is uniform, or if there are many

similar finite elements, there is an advantage to employ the special basis to reduce the number of operations.

The polynomial $p(x)$ can be rewritten as

$$p(x) = \sum_{i=1}^{n} c_i \phi_i(x) + \sum_{i=1}^{m} d_i \psi_i(x).$$

We divide the basis of P_{m+n} into two groups, namely $\{\phi_1, \cdots, \phi_n\}$ and $\{\psi_1, \cdots, \psi_m\}$. Rewriting the matrices Φ, L_Φ and R_Φ defined in Equations (6.5), (6.6) and (6.9) in matrix form as presented in Section 5.6.1, Equation (6.11) can be written as

$$\begin{pmatrix} \Phi^T & \Psi^T \\ L_\Phi^T & L_\Psi^T \end{pmatrix} \begin{pmatrix} \delta c \\ \delta d \end{pmatrix} = \begin{pmatrix} \delta u + r_u \\ -r_N \end{pmatrix}.$$

This is the same equation as Equation (4.24) in Section 4.5. Selecting the special basis such that $\Phi = I$ and $\Psi = O$, for example using the Lagrange basis functions presented in Section 3.6, we find exactly the same equations as Equations (4.25), (4.26) and (4.27).

6.6 Summary of the collocation algorithm

The algorithm of the finite collocation method for nonlinear PDEs is similar to the algorithms presented in Section 4.6 and Section 5.8.

- Mesh generation of 2D domain as presented in Section 5.8.
- Given current approximations to u, v, and c
- Do Newton's iterations until desired the relative errors are smaller than the user prescribed value.
  - Loop over all elements
    - Compute Φ, L_Φ and R_Φ matrices using Equations (6.5), (6.6) and (6.9).
    - Solve Equation (6.12) to find A and B for each element.
    - Find r_u, r_v and r_N, and write Equations (6.14) for each element.
  - Solve the global set of Equation (6.22) for δu and δv. This can be done by the nested dissection algorithm described in the Section 6.4. The itemized steps of this part are presented in Section 4.6.

- For each finite element compute δc using Equation (6.11).
- Update : $u \rightarrow u + \delta u$, $v \rightarrow v + \delta v$, and $c \rightarrow c + \delta c$.
- Calculate the relative errors in the calculation of u, v and c.

- save the results.

Using the Lagrange basis functions as mentioned in Section 6.5, the algorithm of the finite collocation method for nonlinear PDEs changes as follow:

- Mesh generation of 2D domain as presented in Section 5.8.
- Given current approximations to u, v, c and d.
- Do Newton's iterations until desired relative error is achieved.
  - Loop over all elements
    * Compute L_Φ, L_Ψ, R_Ψ and R_Φ matrices using Equations (6.6) and (6.9).
    * Solve Equation (4.26) to find A and B for each element.
    * Find r_u, r_v, and r_N, using Equations (4.28), (4.29) and (6.10), and write Equations (6.14) for each element.
  - Solve the global set of Equation (6.22) for δu and δv. This can be done by the nested dissection algorithm described in the Section 6.4. The itemized steps of this part are presented in Section 4.6.
  - For each finite element compute δc and δd using Equation (4.25).
  - Update : $u \rightarrow u + \delta u$, $v \rightarrow v + \delta v$, $c \rightarrow c + \delta c$, and $d \rightarrow d + \delta d$.
  - Calculate the relative errors in the calculation of u, v, c and d.

- save the results.

Note that as in previous chapters, only one LU-decomposition of the matrix on the left hand side of (6.11), or the matrix L_Ψ in case of using special basis functions, is needed per finite element.

6.7 Stability of the solution algorithm

As the stability of the solution algorithm depends on the stability of the nested dissection mathod, and we use the same nested dissection method as in Section 4.6.1 of Chapter 4, the same arguments in that section are also valid here.

Chapter 7

Continuation of Solutions

Before using the method of finite element collocation with discontinuous piecewise polynomials for continuation problems, we need some general definitions and theorems about continuation problems. In this chapter, we present the continuation of solutions of a nonlinear partial differential equation[1].

Consider the following nonlinear equation

$$N(u, \lambda) = 0, \qquad u,\ N(\cdot, \cdot) \in R^n, \quad \lambda \in R, \tag{7.1}$$

where λ is a control parameter, for example it could be the Reynolds number in a discretized model of a fluid flow problem. In continuation problems, we consider the behavior of solutions as the control parameter changes. If we take $x \equiv (u, \lambda)$, for a shorter notation, the above equation can be rewritten as

$$N(x) = 0, \qquad N : R^{n+1} \rightarrow R^n. \tag{7.2}$$

[1]Sections 7.1 to 7.8 of this chapter are based on reference [47].

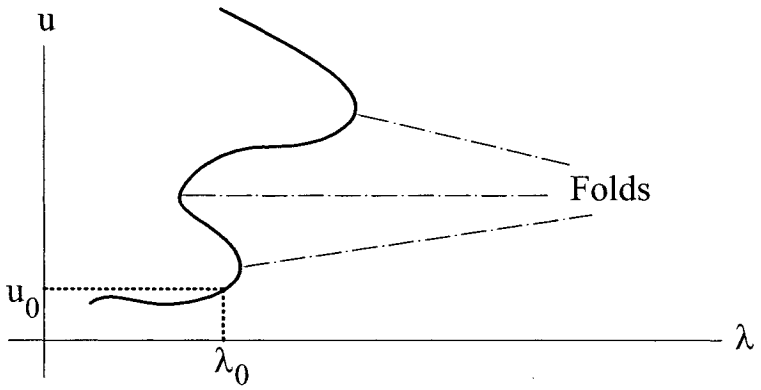


Figure 7.1: Folds on a solution branch

7.1 Regular solutions

A solution x_0 of Equation 7.2 is called a *regular* solution, if $N_x^0 \equiv N_x(x_0)$ has maximal rank, *i.e.*,

$$\text{Rank}(N_x^0) = n.$$

In the parameter formulation, Equation 7.1, N_x^0 is a $n * (n+1)$ matrix,

$$\text{Rank}(N_x^0) \;=\; \text{Rank}(N_u^0 \mid N_\lambda^0) \;=\; n,$$

if and only if either

$$(i)\ N_u^0 \text{ is nonsingular,}$$

or,

$$(ii)\ dim\ \mathcal{N}(N_u^0) = 1, \text{ and } N_\lambda^0 \notin \mathcal{R}(N_u^0).$$

where $\mathcal{N}(.)$ is the null space and $dim\ \mathcal{N}(.)$ is the dimension of the null space. Case (ii) in the above condition is the case of a *simple fold* (see Figure 7.1).

Theorem: 7.1.1 *Take $x_0 \equiv (u_0, \lambda_0)$ as a regular solution of $N(x) = 0$. Then, near x_0, there exists a unique one-dimensional continuum of solutions $x(s)$ with $x(0) = x_0$.*
Proof : *Refer to reference [47].*

7.2 Parameter continuation

Suppose we have a solution (u_0, λ_0) of the nonlinear Equation (7.1), as well as the direction vector $\dot{u}_0$, where $\dot{u} \equiv du/d\lambda$. In the parameter continuation, the

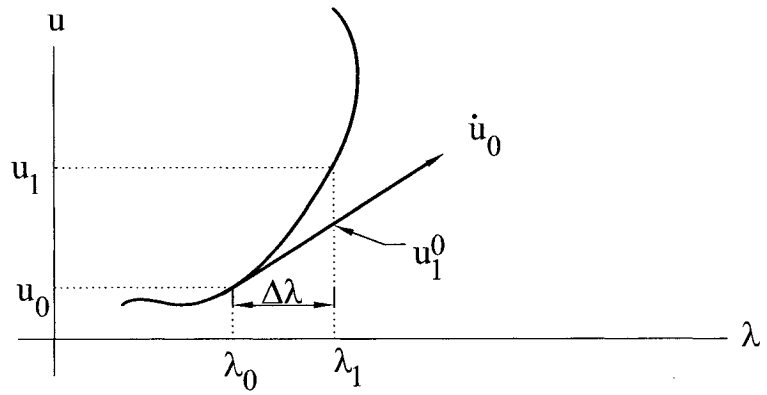


Figure 7.2: Parameter continuation

parameter λ is changed and the solution u_1 at $\lambda_1 \equiv \lambda_0 + \Delta\lambda$ is computed. Figure 7.2 presents the graphical representation of this procedure.

As the equation is a nonlinear one, we use the Newton method.

$$\begin{aligned} N_u(u_1^{(\nu)}, \lambda_1)\ \Delta u_1^{(\nu)} &= -N(u_1^{(\nu)}, \lambda_1), \\ u_1^{(\nu+1)} &= u_1^{(\nu)} + \Delta u_1^{(\nu)}, \end{aligned} \qquad \nu = 0, 1, 2, \cdots, \tag{7.3}$$

where $N_u \equiv dN/du$. As a starting point, we can take $u_1^0 = u_0 + \Delta\lambda\ \dot{u}_0$. According to the convergence theory of Newton's method these iterations converge, if $N_u(u_1, \lambda_1)$ is nonsingular and $\Delta\lambda$ is sufficiently small.

To find the new direction $\dot{u}_1$, the equation $N(u(\lambda), \lambda) = 0$ is differentiated with respect to λ at $\lambda = \lambda_1$.

$$N_u \left.\frac{du}{d\lambda}\right|_{u=u_1, \lambda=\lambda_1} + N_\lambda(u_1, \lambda_1) = 0.$$

As a result, after the convergence of Newton's method, the new direction $\dot{u}_1$ can be obtained by solving the following equation

$$N_u(u_1, \lambda_1)\dot{u}_1 = -N_\lambda(u_1, \lambda_1).$$

As the left hand side of this system is the same as the Newton system, by replacing Δu_1 by $\dot{u}_1$ in Equation (7.3), the calculation of $\dot{u}_1$ can be done without any additional LU factorization of $N_u(u_1, \lambda_1)$.

As parameter continuation cannot pass a fold (see Figure 7.3), we do not use it in this thesis.

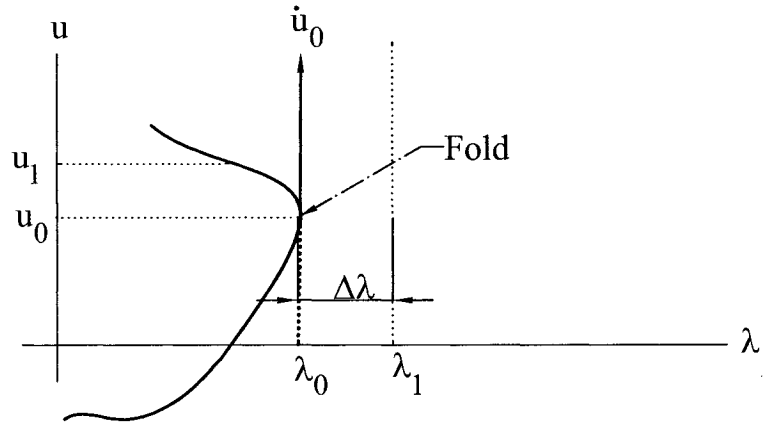


Figure 7.3: Parameter continuation cannot pass a fold

7.3 Keller's pseudo-arclength continuation

To overcome the problem of continuation of solutions past a fold, one can use the pseudo-arclength continuation method introduced by Keller [86]. Suppose we have a solution (u_0, λ_0) of Equation (7.1), as well as the direction vector at the solution point $(\dot{u}_0, \dot{\lambda}_0)$. The pseudo-arclength continuation method consists of introducing a new continuation parameter s and solving the following system for (u_1, λ_1):

$$\begin{aligned} &a) \qquad N(u_1, \lambda_1) = 0, \\ &b) \quad (u_1 - u_0)^T \dot{u}_0 + (\lambda_1 - \lambda_0)\dot{\lambda}_0 - \Delta s = 0, \end{aligned} \tag{7.4}$$

where $\dot{u} \equiv du/ds$ and $\dot{\lambda} \equiv d\lambda/ds$. A graphical interpretation of these equations is presented in Figure 7.4. Actually, Equation (7.4b) represent a step of size Δs in the direction $(\dot{u}_0, \dot{\lambda}_0)$ from the point (u_0, λ_0) to reach the solution point (u_1, λ_1). Of course, the new solution must satisfy nonlinear Equation (7.4a). To solve the above nonlinear system, we use the Newton method

$$\begin{pmatrix} (N_u^1)^{(i)} & (N_\lambda^1)^{(i)} \\ \dot{u}_0^T & \dot{\lambda}_0 \end{pmatrix} \begin{pmatrix} \Delta u_1^{(i)} \\ \Delta\lambda_1^{(i)} \end{pmatrix} = - \begin{pmatrix} N(u_1^{(i)}, \lambda_1^{(i)}) \\ (u_1^{(i)} - u_0)^T \dot{u}_0 - (\lambda_1^{(i)} - \lambda_0)\dot{\lambda}_0 - \Delta s \end{pmatrix}, \tag{7.5}$$

where i is the index of Newton's iteration. After solving the above system the new direction vector can be found by differentiating System (7.4) with respect to s:

$$\begin{pmatrix} N_u^1 & N_\lambda^1 \\ \dot{u}_0^T & \dot{\lambda}_0 \end{pmatrix} \begin{pmatrix} \dot{u}_1 \\ \dot{\lambda}_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{7.6}$$

The new direction vector is rescaled, so that $\| \dot{u}_1 \|^2 + \dot{\lambda}_1^2 = 1$.

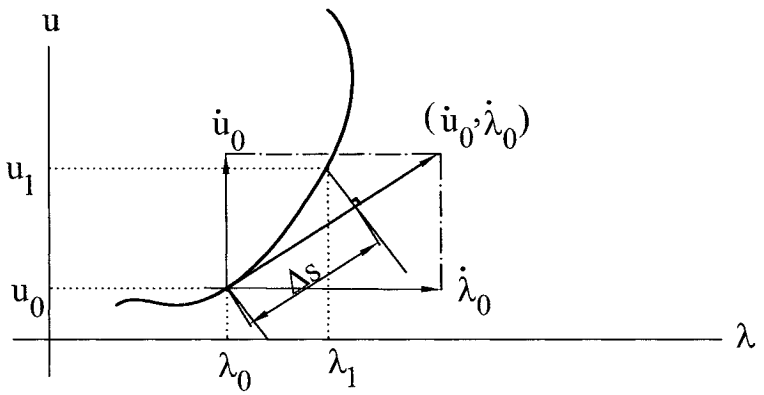


Figure 7.4: Pseudo-arclength continuation

The new direction vector $(\dot{u}_1, \dot{\lambda}_1)$ can be calculated with just one extra back-substitution of the Newton system with the new right hand side of Equation (7.6). The orientation of the branch is also preserved if Δs is sufficiently small.

Theorem: 7.3.1 *The Jacobian of the pseudo-arclength system is nonsingular at a regular solution point.*

Proof : *Refer to reference [47].*

7.4 Simple folds

The solution point (u_0, λ_0) of Equation (7.1) is called *a simple fold*, if we have

$$\mathcal{N}(N_u^0) = \text{Span}\{\phi\},$$

$$\mathcal{N}(N_u^{0^T}) = \text{Span}\{\psi\},$$

and

$$N_\lambda^0 \notin \mathcal{R}(N_u^0),$$

where $N_u^0 \equiv N_u(u_0, \lambda_0)$, and $\|\phi\| = \|\psi\| = 1$.

In other word, a regular solution $x_0 \equiv (u_0, \lambda_0)$ of Equation (7.1) is called a simple fold, if

$$dim\ \mathcal{N}(N_u^0) = 1,$$

and

$$N_\lambda^0 \notin \mathcal{R}(N_u^0).$$

If we differentiate $N(u(s), \lambda(s)) = 0$ with respect to the continuation parameter s we have

$$N_u(u(s), \lambda(s))\dot{u}(s) + N_\lambda(u(s), \lambda(s))\dot{\lambda}(s) = 0, \tag{7.7}$$

where $\dot{u} = du/ds$ and $\dot{\lambda} = d\lambda/ds$. This equation can be written for the point (u_0, λ_0) as

$$N_u^0 \dot{u}_0 = -\dot{\lambda}_0 N_\lambda^0 \ .$$

At a fold $N_\lambda^0 \notin \mathcal{R}(N_u^0)$, thus $\dot{\lambda}_0 = 0$. As a result

$$N_u^0 \dot{u}_0 = 0.$$

Since $dim\ \mathcal{N}(N_u^0) = 1$, we have

$$\mathcal{N}(N_u^0) = \text{Span}\{\dot{u}_0\}.$$

Differentiate Equation (7.7) for the second time, we have

$$N_u^0 \ddot{u}_0 + N_\lambda^0 \ddot{\lambda}_0 + N_{uu}^0 \dot{u}_0 \dot{u}_0 + 2N_{u\lambda}^0 \dot{u}_0 \dot{\lambda}_0 + N_{\lambda\lambda}^0 \dot{\lambda}_0 \dot{\lambda}_0 = 0. \tag{7.8}$$

Take $\phi = \dot{u}_0$, then at a simple fold (u_0, λ_0) we can write

$$\mathcal{N}(N_u^0) = \text{Span}\{\phi\},$$

and

$$\mathcal{N}((N_u^0)^T) = \text{Span}\{\psi\}.$$

Multiply Equation (7.8) by ψ^T and use the facts that $\dot{\lambda}_0 = 0$ and $\psi \perp \mathcal{R}(N_u^0)$ we find

$$\psi^T N_\lambda^0 \ddot{\lambda}_0 + \psi^T N_{uu}^0 \phi\phi = 0.$$

Here $\psi^T N_\lambda^0 \neq 0$ since $N_\lambda^0 \notin \mathcal{R}(N_u^0)$. Thus

$$\ddot{\lambda}_0 = \frac{-\psi^T N_{uu}^0 \phi\phi}{\psi^T N_\lambda^0}.$$

If the curvature $\ddot{\lambda}_0 \neq 0$, then (u_0, λ_0) is called a *simple quadratic fold.*

7.5 Simple singular points

Suppose $x_0 \equiv (u_0, \lambda_0)$ is a solution of nonlinear Equation (7.1) or (7.2). The point x_0 is called a *simple singular point* if we have

$$\mathcal{N}(N_x^0) = \text{Span}\{\phi_1, \phi_2\}, \quad \mathcal{N}(N_x^{0^T}) = \text{Span}\{\psi\}. \tag{7.9}$$

This is a singular case that, in practice, occurs along a solution branch for many nonlinear equations. Typically, this case corresponds to intersecting solution branches, *i.e.*, a bifurcation. The linear operator N_x^0 is a matrix with n rows and $n+1$ columns. The rank of such matrix is at most n. The above condition means that the rank of this matrix is $n-1$.

In the parameter formulation, where $N_x^0 = (N_u^0 \mid N_\lambda^0)$, we have $x_0 = (u_0, \lambda_0)$ is a simple singular point if and only if

- $dim\ \mathcal{N}(N_u^0) = 1, \qquad N_\lambda^0 \in \mathcal{R}(N_u^0),$

or

- $dim\ \mathcal{N}(N_u^0) = 2, \qquad N_\lambda^0 \notin \mathcal{R}(N_u^0).$

Suppose we have a solution branch $x(s)$ of $N(x) = 0$, where s is some parametrization. Assume $x_0 \equiv (u_0, \lambda_0)$ is a simple singular point. Then we must have

$$\begin{aligned}
&N(x(s)) = 0,\\
&N^0 = N(x_0) = 0,\\
&N_x(x(s))\dot{x}(s) = 0,\\
&N_x^0\dot{x}_0 = N_x(x_0)\dot{x}_0 = 0,\\
&N_{xx}(x(s))\dot{x}(s)\dot{x}(s) + N_x(x(s))\ddot{x}(s) = 0,\\
&N_{xx}^0\dot{x}_0\dot{x}_0 + N_x^0\ddot{x}_0 = 0.
\end{aligned}$$

Take $\dot{x}_0 = \alpha\phi_1 + \beta\phi_2$, for some $\alpha, \beta \in R$, then we can write

$$\psi^T N_{xx}^0(\alpha\phi_1 + \beta\phi_2)(\alpha\phi_1 + \beta\phi_2) + \psi^T N_x^0\ddot{x}_0 = 0.$$

As $\psi^T N_x^0 = 0$, we have

$$(\psi^T N_{xx}^0\phi_1\phi_1)\alpha^2 + 2(\psi^T N_{xx}^0\phi_1\phi_2)\alpha\beta + (\psi^T N_{xx}^0\phi_2\phi_2)\beta^2 = 0. \tag{7.10}$$

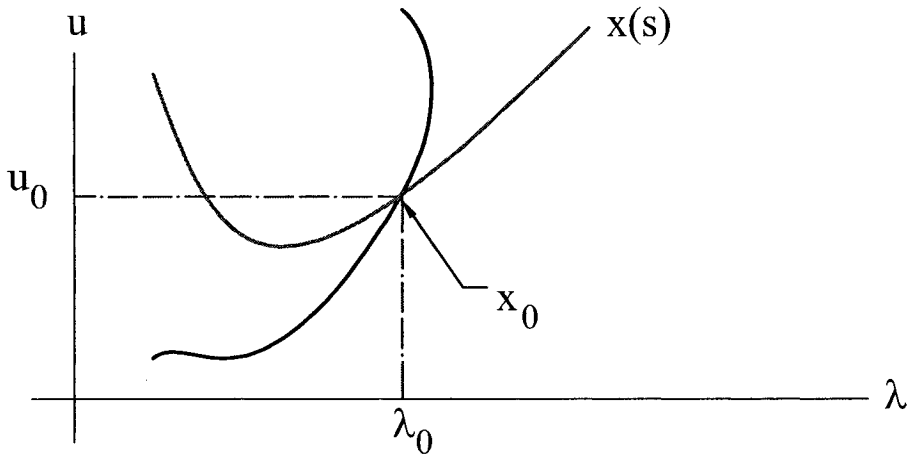


Figure 7.5: A branch point x_0 on a solution branch

Equation (7.10) is called the *Algebraic Bifurcation Equation* (ABE). We must solve this equation for (α, β) pairs without both α and β equal to zero. Take

$$\psi^T N^0_{xx} \phi_1 \phi_1 = c_{11},$$

$$\psi^T N^0_{xx} \phi_1 \phi_2 = c_{12},$$

and

$$\psi^T N^0_{xx} \phi_2 \phi_2 = c_{22},$$

then Equation (7.10) can be written as

$$c_{11}\alpha^2 + 2c_{12}\alpha\beta + c_{22}\beta^2 = 0. \tag{7.11}$$

If the *discriminant*

$$\Delta \equiv c_{12}^2 - c_{11}c_{22} \; > \; 0, \tag{7.12}$$

then the ABE has two distinct real nontrivial solution pairs (α_1, β_1) and (α_2, β_2), which are unique up to scaling. In such case we have a *bifurcation*, (or *branch point*) *i.e.*, two distinct branches pass through x_0. A graphical interpretation is presented in Figure 7.5.

7.6 Computing the bifurcation direction

For a bifurcating branch, the equation of the direction vector can be written as

$$\dot{x}_0 = \alpha\phi_1 + \beta\phi_2,$$

where (α, β) is a root of the algebraic bifurcation equation (7.11). At the branch point, assume ϕ_1 is the direction of the current branch, *i.e.*, $\phi_1 = \dot{x}_0$. Therefore $(\alpha_1, \beta_1) = (1, 0)$ must be one of roots of Equation (7.11). As a result, we have

$$c_{11} = 0.$$

Assuming $\Delta > 0$, thus we have $c_{12} \neq 0$. Then the second root must satisfy the following equation

$$\alpha_2/\beta_2 = -c_{22}/2c_{12}.$$

The null vectors of the system can be found as follow

ϕ_1 : Already selected as $\phi_1 = \dot{x}_0$.

ϕ_2 : As we have $\phi_2 \perp \phi_1$. Then ϕ_2 is a null vector of $F_x^0 = \begin{pmatrix} N_x^0 \\ \dot{x}_0^T \end{pmatrix}$.

ψ : $\begin{pmatrix} \psi \\ 0 \end{pmatrix}$ is a simple null vector of $(F_x^0)^T = ((N_x^0)^T \mid \dot{x}_0)$.

F_x^0 is the Jacobian of the extended pseudo-arclength system at x_0 (see Section 7.3). We need this null vectors to evaluate c_{12} and c_{22}. The null space of F_x^0 is $1D$ as will be shown in Section 7.8[2].

Thus the direction vector of the bifurcating branch is

$$x_0' \equiv \alpha_2\phi_1 + \beta_2\phi_2.$$

We scale the direction vector after determining the coefficients α_2 and β_2 such that $\| x_0' \| = 1$.

7.7 Switching branches

Using Keller's pseudo-arclength continuation, Section 7.3, the first solution point x_1 on the bifurcating branch can be computed as follow

$$\begin{aligned} N(x_1) &= 0, \\ (x_1 - x_0)^T x_0' - \Delta s &= 0, \end{aligned} \tag{7.13}$$

[2]Left and right null vectors of a matrix can be computed at little extra cost once the matrix has been LU decomposed.

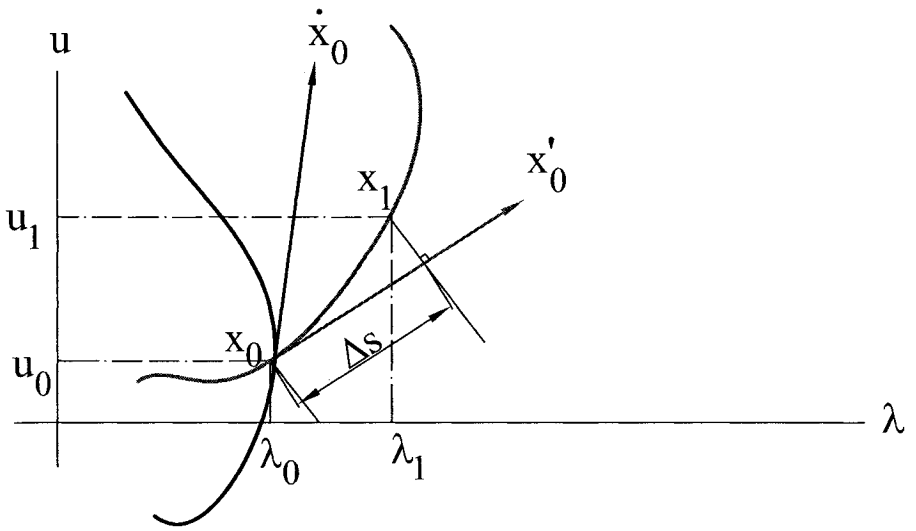


Figure 7.6: Switching branches using the correct bifurcation direction

where x_0 is the branch point, and x'_0 is the direction of the bifurcating branch. We can use the following initial approximation to the point x_1

$$x_1^{(0)} = x_0 + \Delta s \; x'_0.$$

Figure 7.6 presents a graphical interpretation of the procedure. From Section 7.6, one conclude that for the computation of x'_0 the evaluation of N^0_{xx} is needed.

A simplified branch switching procedure is also possible. To find the first solution point on the bifurcating branch, instead of solving System (7.13), we can solve the following equations

$$N(x_1) = 0,$$
$$(x_1 - x_0)^T \phi_2 - \Delta s = 0.$$

where ϕ_2 is the second null vector of N^0_x. As mentioned in Section 7.6, we have

$$\phi_2 \perp \phi_1,$$

where $\phi_1 = \dot{x}_0$. Figure 7.7 shows a graphical interpretation of this method.

We can possibly find situations where this method fails, but it works well in almost all practical applications. Using this method, we do not need to evaluate the second derivatives of N (*i.e.*, N^0_{xx}) as needed in the original method. We implemented this simplified branch switching procedure in our prototype software.

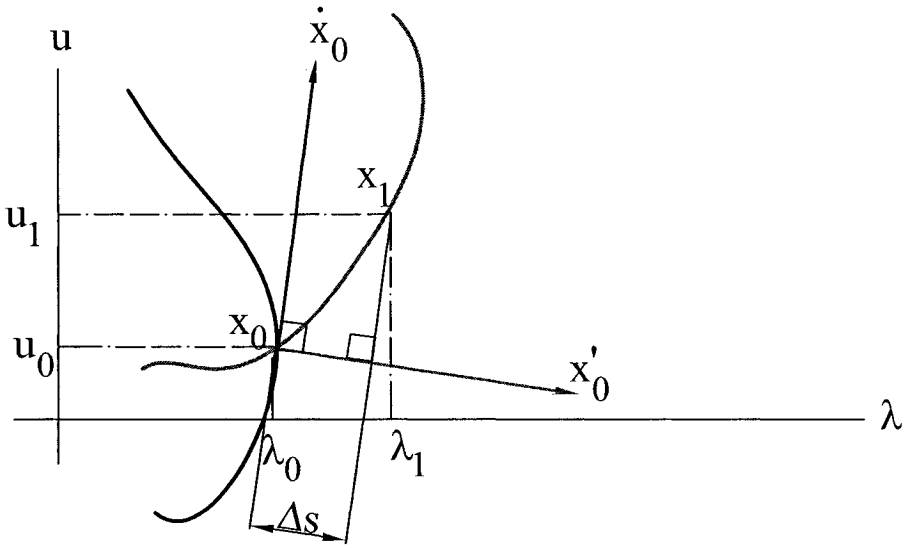


Figure 7.7: Switching branches using the orthogonal direction

7.8 Detection of bifurcation points

The pseudo-arclength equation (7.4) of Section 7.3 can be written as

$$F(x;s) \equiv \begin{pmatrix} N(x) \\ \\ (x-x_0)^T \dot{x}_0 - \Delta s \end{pmatrix}. \tag{7.14}$$

If x_0 is a simple singular point. Then we have

$$F_x^0 \equiv F_x(x_0;s_0) = F_x(x_0) = \begin{pmatrix} N_x^0 \\ \\ \dot{x}_0^T \end{pmatrix}. \tag{7.15}$$

Using the discussion of Section 7.6, we can take $\phi_1 = \dot{x}_0$ as the first null vector of N_x^0. Equation (7.15) can be written as

$$F_x^0 = \begin{pmatrix} N_x^0 \\ \\ \phi_1^T \end{pmatrix}. \tag{7.16}$$

For the second null vector of N_x^0, we choose ϕ_2 such that $\phi_2^T \phi_1 = 0$. Then ϕ_2 is also a null vector of F_x^0, but ϕ_1 is not. As a result, F_x^0 has a $1D$ null space. Since ψ is the null vector of N_x^{0T}, we have that F_x^{0T} has the following the null vector

$$\Psi \equiv \begin{pmatrix} \psi \\ 0 \end{pmatrix}.$$

Theorem: 7.8.1 *Let $x_0 = x(s_0)$ be a simple singular point on a smooth solution branch $x(s)$ of $N(x) = 0$. Let $F(x;s)$ defined by Equation (7.14). Assume that the discriminant Δ defined by Equation (7.12) is positive, and Equation (7.15) has 0 as an algebraically simple eigenvalue. Then the determinant of F_x changes sign at x_0.*

Proof : *Consider the following eigenvalue problem*

$$F_x(x(s))\ \phi(s) \ = \ \kappa(s)\ \phi(s), \tag{7.17}$$

where $\kappa(s)$ and $\phi(s)$ are smooth near s_0, and

$$\kappa(s_0) = 0 \quad and \quad \phi(0) = \phi_2,$$

i.e., the pair $(\kappa(s), \phi(s))$ is the continuation of $(0, \phi_2)$. This can be done because we have assumed that 0 is an algebraically simple eigenvalue. We differentiate Equation (7.17), and we evaluate it at s_0, using $\kappa(0) = 0$, $\dot{x}_0 = \phi_1$ and $\phi(0) = \phi_2$. This gives

$$F^0_{xx}\phi_1\phi_2 + F^0_x\dot{\phi}(s_0) = \dot{\kappa}_0\phi_2.$$

If we multiply the above equation on the left by Ψ^T, we find

$$\dot{\kappa}_0 = \frac{\Psi^T F^0_{xx}\phi_1\phi_2}{\Psi^T\phi_2} = \frac{(\psi^T, 0)\begin{pmatrix} N^0_{xx} \\ 0 \end{pmatrix}\phi_1\phi_2}{\Psi^T\phi_2} = \frac{\psi^T N^0_{xx}\phi_1\phi_2}{\Psi^T\phi_2} = \frac{c_{12}}{\Psi^T\phi_2}.$$

Note that the left and right null vectors cannot be orthogonal, i.e.,

$$\Psi^T\phi_2 \neq 0.$$

Now c_{12} is a coefficient of the ABE, and as we assume that $\Delta \neq 0$, it follows that $c_{12} \neq 0$. Thus $\dot{\kappa}_0 \neq 0$.

The following theorem, which we present without proof[3], states that there must be a bifurcation point at x_0 if the determinant of the pseudo-arclength Jacobian matrix changes sign.

[3]This theorem can be proven by degree theory.

Theorem: 7.8.2 *Consider $x(s)$ be a smooth solution branch of*

$$F(x; s) = 0,$$

where

$$F : R^{n+1} \times R \rightarrow R^{n+1},$$

with a C^1 continuity. Assume that the determinant of $F_x(x(s); s)$ changes sign at $s = s_0$. Then $x(s_0)$ is a bifurcation point, i.e., every open neigborhood of x_0 contains a solution of $F(x; s) = 0$ that does not lie on $x(s)$.

In our software, the detection of simple singular points is related to the above theorem. One monitors the sign of the determinant of the pseudo-arclength Jacobian matrix, Equation (7.5), in each continuation step. If the sign changes between two solution points then an iterative method such as the Regula-Falsi, the bisection method or Müller's method (with bracketing) can be used to accurately locate the singular point. The Regula-Falsi and the bisection methods are implemented in our software.

As we use the modified nested dissection method, where at each interior node of the binary tree a pivoting strategy is used for the union of two subregions, a sign change of the determinant can occur without passing a branch point. Therefore, we actually monitor the absolute value of the determinant. If in an interval this value decreases and then increases, or it increases and then decreases, we fix the pivoting, and we monitor the sign change of the determinant on that interval. In large systems, to avoid overflow, one can compute a scaled determinant.

7.9 Modified pseudo-arclength continuation

When the nonlinear system (7.1) strongly depends on a control parameter, for example, systems such as the Brusselator system (see Section 11.6), where boundary conditions depend on the control parameter, it is difficult to detect, and to locate, branch points using the Newton system (7.5). In this case the locating algorithm, based on the Regula-Falsi or the bisection method, converges slowly or it does not locate the branch point at all. This problem is due to the way that one proceeds from point X_0 to X_1. When the Newton method converges at point X_1 the determinant of

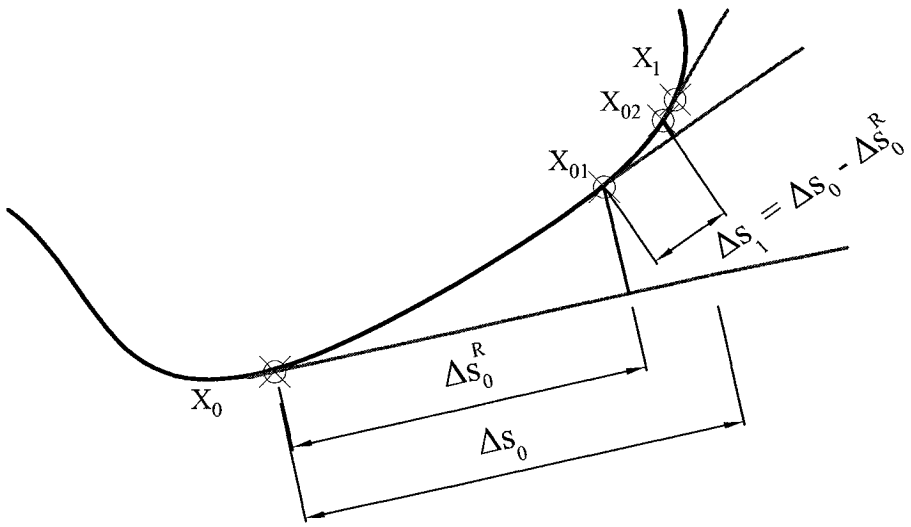


Figure 7.8: Modified pseudo-arclength with a variable direction vector

System (7.5) does not present correctly the system at point X_1, because the direction vector is belong to the point X_0. The determinant of this system is not suitable for detecting and locating the bifurcation point between X_0 and X_1.

More precisely, in the first Newton's iteration, using the point X_0 as a first guess to the solution, after solving System (7.5) one finds $\Delta X = (\Delta u, \Delta\lambda)$, which identifies the point $X_{01} = X_0 + \Delta X$ (see Figure 7.8). Note that System (7.5) is evaluated for the point X_0 not X_{01}. First, because the direction vector is not the same at these two points. Second, the determinant of the system is evaluated at X_0, but we are at X_{01}, *i.e.*, we are at point X_{01} but the information belongs to a different point. To overcome this problem we propose some modifications to the original pseudo-arclength method.

First, we must reduce Δs after each Newton's iteration, such that the point X_1 is approached during the Newton's iteration. Second, we update the direction vector in each Newton's iteration. Figure 7.8 illustrates this procedure. As one can see, after each iteration, we reduce Δs. For example, after the first iteration with step size equal to Δs_0, we reach the point X_{01}, at distance Δs_0^R from X_0 (on the line of the direction vector). Thus, for the second iteration we have $\Delta s_1 = \Delta s_0 - \Delta s_0^R$. For iteration i we can write

$$\Delta s_i = \Delta s_{i-1} - \Delta s_{i-1}^R.$$

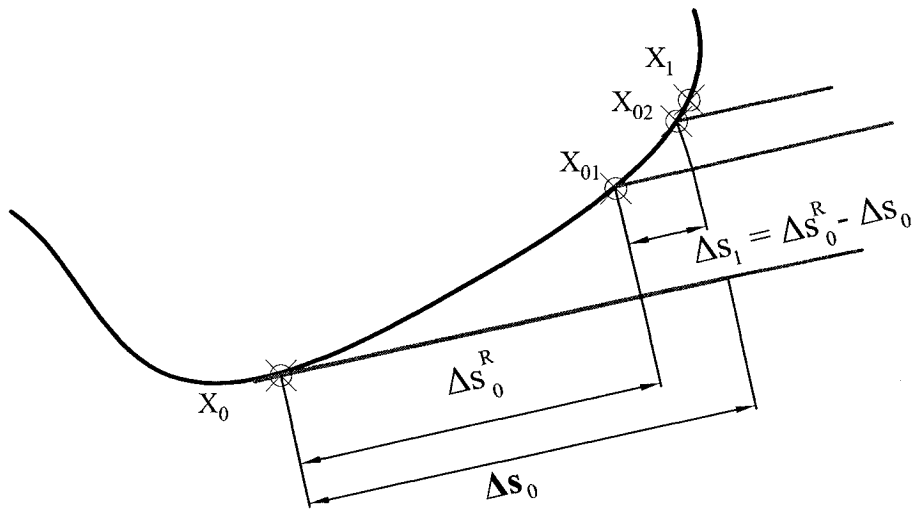


Figure 7.9: Modified pseudo-arclength with a constant direction vector

Then Equation (7.5) changes as follows

$$\begin{pmatrix} (N_u^1)^{(i)} & (N_\lambda^1)^{(i)} \\ \dot{u}_i^T & \dot{\lambda}_i \end{pmatrix} \begin{pmatrix} \Delta u_1^{(i)} \\ \Delta \lambda_1^{(i)} \end{pmatrix} = - \begin{pmatrix} N(u_1^{(i)}, \lambda_1^{(i)}) \\ (u_1^{(i)} - u_i)^T \dot{u}_i - (\lambda_1^{(i)} - \lambda_i)\dot{\lambda}_i - \Delta s_i \end{pmatrix}, \quad (7.18)$$

where i is the index of Newton's iteration. Usually changing the step size Δs in each Newton's iteration is enough, and converge is better for some applications. The graphical representation of continuation without changing the direction vector is presented in Figure 7.9. As one can see, after each Newton's iteration Δs changes, but the direction vector stays the same.

Chapter 8

The Collocation Method for Continuation Problems in Nonlinear Elliptic PDEs

8.1 Introduction

In this chapter, we use the collocation method presented in Chapters (5) and (6), *i.e.*, the discontinuous piecewise polynomial collocation method, for the numerical continuation of solutions in nonlinear elliptic PDEs.

8.2 Collocation with discontinuous piecewise polynomials

Consider the following scalar parameter-dependent nonlinear elliptic PDE

$$Nu(x) \equiv \Delta u(x) + f(u(x), \lambda) = 0, \quad \text{for} \quad x \in \Omega, \tag{8.1}$$

with the boundary condition

$$u(x) = 0, \quad \text{for} \quad x \in \partial\Omega, \tag{8.2}$$

where Δ is the Laplace operator, Ω the unit square in R^2, $\lambda \in R$, $x = (x_1, x_2)^T$, the superscript ^{T} denotes the transpose operator, and $f(u(x), \lambda), u(x) \in R$.

In the continuation algorithm for parameter-dependent problems, in addition to $u(\cdot)$, we also treat λ as unknown. For using the Keller's pseudo-arclength continuation method presented in Chapter 7, we add an integral constraint of the form

$$\int_\Omega q(x, u(x), \lambda) = 0, \tag{8.3}$$

where $q(\cdot) \in R$.

As mentioned before, our collocation method and the nested dissection procedure can be generalized to problems with $x \in R^n$. Furthermore, the domain need not be a square, and more general elements are possible. More general boundary conditions and multiple integral constraints can also be treated. However, in order to keep the technical presentation simple, we restrict attention in this chapter to the problem presented in Equations (8.1) to (8.3).

To generate a suitable mesh, as in Chapter 6, the square domain Ω is recursively subdivided into 2^M finite elements, where M is the level of subdivision in the binary tree, refer to Figure 5.1. For each finite element, we select appropriate boundary matching points x_i, $i = 1, \cdots, n$, and interior collocation points z_i, $i = 1, \cdots, m$. For an example see Figure 5.2. As before, the two sets of matching points on the common boundary of two adjacent elements must coincide. A polynomial $p(x) \in P_{n+m}$ is associated with each finite element, where P_{n+m} is an appropriate $(n+m)$-dimensional polynomial space. At the matching points the values of the neighboring polynomials u_i are required to match. The normal derivatives of the neighboring polynomials v_i are also required to match at these points. Further, for each element, the associated polynomial must satisfy the collocation equations

$$Np(z_i) \equiv \Delta p(z_i) + f\Big(p(z_i), \lambda\Big) = 0, \qquad i = 1, \cdots, m. \tag{8.4}$$

We suppress the index of the element, for notational simplicity. Finally, discrete, boundary conditions are imposed at points x_i on the domain boundary $\partial\Omega$. For each finite element the local polynomial has the form $p(x) = \sum_{i=1}^{n+m} c_i \phi_i(x)$, where $Span\Big\{\phi_1, \cdots, \phi_{n+m}\Big\} = P_{n+m}$. Then the collocation equations are

$$Np(z_k) \equiv \sum_{i=1}^{n+m} c_i \Delta\phi_i(z_k) + f\Big(\sum_{i=1}^{n+m} c_i \phi_i(z_k), \lambda\Big) = 0, \qquad k = 1, \cdots, m. \tag{8.5}$$

As in previous chapters, to impose the continuity requirements, we associate unique variables u_i and v_i to each matching point x_i, namely, we require that $p(x_i) = u_i$ and $\nabla p(x_i)^T \eta_i = v_i$. We assume that for any given x_i the outward normal η_i has a unique orientation. Take $u = (u_1, \cdots, u_n)^T$, $v = (v_1, \cdots, v_n)^T$, and

$$\Phi = \begin{pmatrix} \phi_1(x_1) & \cdots & \phi_1(x_n) \\ \vdots & & \vdots \\ \phi_{n+m}(x_1) & \cdots & \phi_{n+m}(x_n) \end{pmatrix},$$

$$R_\Phi = \begin{pmatrix} \nabla\phi_1(x_1)^T\eta_1 & \cdots & \nabla\phi_1(x_n)^T\eta_n \\ \vdots & & \vdots \\ \nabla\phi_{n+m}(x_1)^T\eta_1 & \cdots & \nabla\phi_{n+m}(x_n)^T\eta_n \end{pmatrix}.$$

Then we can write

$$u - \Phi^T c = 0, \tag{8.6}$$

$$v - R_\Phi^T c = 0. \tag{8.7}$$

The integral constraint (8.3) is applied to the union of all local polynomials. This equation can be written as

$$\sum_{\ell=1}^{2^M} \int_{\Omega_\ell} q\big(x, p(x), \lambda\big) d\Omega = 0, \tag{8.8}$$

where Ω_ℓ denotes the ℓth finite element of Ω. As an example, we can integrate over each element by an appropriate quadrature formula that uses the collocation points z_k, with weights ω_k. In such case we have

$$\sum_{\ell=1}^{2^M} \sum_{k=1}^{m} \omega_k q\big(z_k, p(z_k), \lambda\big) = 0. \tag{8.9}$$

8.3 Newton's method

Equations (8.5), (8.6), (8.7) and (8.8), together with the discrete boundary conditions, constitute the discretization. The unknowns are $c \in R^{n+m}$ for each finite element, the u_i and v_i associated with the points x_i on the inter-element boundaries, and the control parameter λ. To solve these equations, we use Newton's method. Omitting iteration indices, it can now be written as

$$L_\Phi^T \, \delta c + \delta\lambda \, f_\lambda = -r_N, \tag{8.10}$$

$$\delta u - \Phi^T \delta c = -r_u, \tag{8.11}$$

$$\delta v - R_\Phi^T \delta c = -r_v. \tag{8.12}$$

and

$$\sum_{\ell=1}^{2^M} \Big\{ q_u^T \ \delta c + \delta\lambda \ q_\lambda + r_q \Big\} = 0. \tag{8.13}$$

Above

$$L_\Phi^T = \begin{pmatrix} L\big[p(z_1)\big]\phi_1(z_1) & \cdots & L\big[p(z_1)\big]\phi_{n+m}(z_1) \\ \vdots & & \vdots \\ L\big[p(z_m)\big]\phi_1(z_m) & \cdots & L\big[p(z_m)\big]\phi_{n+m}(z_m) \end{pmatrix},$$

$$f_\lambda = \begin{pmatrix} D_2 f\big(p(z_1), \lambda\big) \\ \vdots \\ D_2 f\big(p(z_m), \lambda\big) \end{pmatrix},$$

where, as before, L is the linearization of N, *i.e.*, $L[p(z)]\phi(z)$ is the linearization of N about p acting on ϕ at z, or,

$$L[p(z)]\phi(z) = \Delta\phi(z) + D_1 f\big(p(z), \lambda\big)\phi(z).$$

$D_1 f(.,.)$ is the derivative with respect to the first argument and $D_2 f(.,.)$ is the derivative with respect to the second argument. Further, we have defined

$$\delta c = \begin{pmatrix} \delta c_1 \\ \vdots \\ \delta c_{n+m} \end{pmatrix}, \quad r_N = \begin{pmatrix} Np(z_1) \\ \vdots \\ Np(z_m) \end{pmatrix},$$

$$\delta u = \begin{pmatrix} \delta u_1 \\ \vdots \\ \delta u_n \end{pmatrix}, \quad \delta v = \begin{pmatrix} \delta v_1 \\ \vdots \\ \delta v_n \end{pmatrix},$$

$$r_u = u - \Phi^T c, \qquad r_v = v - R_\Phi^T c,$$

and

$$q_u = \int_{\Omega_\ell} \begin{pmatrix} D_2 q\big(x, p(x), \lambda\big)\phi_1(x) \\ \vdots \\ D_2 q\big(x, p(x), \lambda\big)\phi_{n+m}(x) \end{pmatrix} d\Omega,$$

$$q_\lambda = \int_{\Omega_\ell} D_3 q\big(x, p(x), \lambda\big) d\Omega,$$

$$r_q = \int_{\Omega_\ell} q\big(x, p(x), \lambda\big) d\Omega.$$

Using an appropriate quadrature formula, we have

$$q_u = \begin{pmatrix} \sum_{k=1}^m \omega_k D_2 q\big(z_k, p(z_k), \lambda\big)\phi_1(z_k) \\ \vdots \\ \sum_{k=1}^m \omega_k D_2 q\big(z_k, p(z_k), \lambda\big)\phi_{n+m}(z_k) \end{pmatrix},$$

$$q_\lambda = \sum_{k=1}^{m} \omega_k D_3 q\Big(z_k, p(z_k), \lambda\Big),$$

$$r_q = \sum_{k=1}^{m} \omega_k q\Big(z_k, p(z_k), \lambda\Big).$$

Equations (8.10) and (8.11) can be written as

$$\begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix} \delta c = \begin{pmatrix} \delta u + r_u \\ -r_N - f_\lambda \delta\lambda \end{pmatrix}. \tag{8.14}$$

Using Equation (8.14) to eliminate δc in Equation (8.12) one obtains

$$\delta v = R_\Phi^T \begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix}^{-1} \begin{pmatrix} \delta u + r_u \\ -r_N - f_\lambda \delta\lambda \end{pmatrix} - r_v.$$

Define A and B, as in Section 6.3,

$$(\Phi \mid L_\Phi) \begin{pmatrix} A^T \\ B^T \end{pmatrix} = R_\Phi. \tag{8.15}$$

Then the above expression for δv can be rewritten as

$$\delta v = (A \quad B) \begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix} \begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix}^{-1} \begin{pmatrix} \delta u + r_u \\ -r_N - f_\lambda \delta\lambda \end{pmatrix} - r_v,$$

that is,

$$\delta v = A\delta u - \delta\lambda B\ f_\lambda - B r_N - r_v + A r_u.$$

This equation is of the form

$$\delta v = A\delta u + \delta\lambda\ b + r, \tag{8.16}$$

where

$$r = -B r_N - r_v + A r_u, \qquad b = -B f_\lambda.$$

Using Equation (8.14) and Equation (8.13), we can write

$$\sum_{\ell=1}^{2^M} \Big\{ q_u^T \begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix}^{-1} \begin{pmatrix} \delta u + r_u \\ -r_N - f_\lambda \delta\lambda \end{pmatrix} + \delta\lambda\ q_\lambda + r_q \Big\} = 0. \tag{8.17}$$

Take $d \in R^n$, $\hat{d} \in R^m$, to be the solution of

$$(\Phi \mid L_\Phi) \begin{pmatrix} d \\ \hat{d} \end{pmatrix} = q_u. \tag{8.18}$$

Then Equation (8.17) can be written as

$$\sum_{\ell=1}^{2^M} \Big\{ d^T(\delta u + r_u) + \hat{d}^T(-r_N - f_\lambda \delta\lambda) + \delta\lambda\ q_\lambda + r_q \Big\} = 0,$$

which is of the form

$$\sum_{\ell=1}^{2^M} \{d^T \delta u + e\ \delta\lambda + s\} = 0, \tag{8.19}$$

where

$$e = q_\lambda - \hat{d}^T f_\lambda,$$

$$s = r_q + d^T r_u - \hat{d}^T r_N.$$

The Newton equations for the boundary conditions, to be applied at the points on the exterior boundary $\delta\Omega$, can be written as

$$\delta u = -u. \tag{8.20}$$

We can exclude the boundary unknowns in case of explicit boundary conditions for u. However, including Equation (8.20) in the solution algorithm has an advantage for the treatment of more general boundary conditions.

8.4 Nested dissection

In this section, we consider the application of the nested dissection method presented in Sections 6.4 to the solution of the global set of Equations (8.16) and (8.19). The algorithm, as before, consists of a backward recursive elimination of the unknowns δu and δv on boundaries separating adjacent regions. Note again that the backward recursion means the elimination starts at the leaves and terminates at the root of the binary tree. This procedure results in the elimination of all the δu's and δv's in the interior of the domain Ω. One is left with one equation corresponding to the matching point on the exterior boundary $\delta\Omega$. Then, the discrete boundary conditions and the transformed integral constraint, Equation (8.19), can be used to determine $\delta\lambda$ and the values of δu, δv at the points x_i on $\delta\Omega$. Thereafter a recursive back-substitution gives the values of δu and δv on each interior matching point.

Take

$$g = \delta\lambda\ b + r, \tag{8.21}$$

then Equation (8.16) can be written in the form of Equation (6.14), namely

$$D\delta v = A\delta u + g, \tag{8.22}$$

As in Section 6.4 the matrix D is an identity matrix at the element level. Take two adjacent regions Ω_1 and Ω_2 as in Figure 5.4. Equation (8.22) can be written in the split form for regions Ω_1 and Ω_2.

$$\begin{pmatrix} D_{11}^1 & D_{12}^1 \\ D_{21}^1 & D_{22}^1 \end{pmatrix} \begin{pmatrix} \delta v_1 \\ \delta v_{12} \end{pmatrix} = \begin{pmatrix} A_{11}^1 & A_{12}^1 \\ A_{21}^1 & A_{22}^1 \end{pmatrix} \begin{pmatrix} \delta u_1 \\ \delta u_{12} \end{pmatrix} + \begin{pmatrix} g_1^1 \\ g_{12}^1 \end{pmatrix}, \tag{8.23}$$

$$\begin{pmatrix} D_{11}^2 & D_{12}^2 \\ D_{21}^2 & D_{22}^2 \end{pmatrix} \begin{pmatrix} \delta v_2 \\ \delta v_{21} \end{pmatrix} = \begin{pmatrix} A_{11}^2 & A_{12}^2 \\ A_{21}^2 & A_{22}^2 \end{pmatrix} \begin{pmatrix} \delta u_2 \\ \delta u_{21} \end{pmatrix} + \begin{pmatrix} g_2^2 \\ g_{21}^2 \end{pmatrix}, \tag{8.24}$$

where

$$g_1^1 = \delta\lambda\ b_1^1 + r_1^1,$$

$$g_{12}^1 = \delta\lambda\ b_{12}^1 + r_{12}^1,$$

$$g_2^2 = \delta\lambda\ b_2^2 + r_2^2,$$

$$g_{21}^2 = \delta\lambda\ b_{21}^2 + r_{21}^2.$$

The superscript represents the region number. Equations (8.23) and (8.24) are the same as Equations (6.15) and (6.16), and matching conditions between adjacent regions are also the same as in Chapter 6. Thus the elimination of the unknowns δu and δv on the common boundary, based on Equation (8.16) or its equivalent Equation (8.22), is the same as the procedure described in Section 6.4, and we omit repeating the details here.

Equation (8.19) can be written as

$$\sum_{\ell=1}^{2^M} \{d^T \delta u + t^T \delta v + e\ \delta\lambda + s\} = 0, \tag{8.25}$$

where t is a zero vector at the element level, but not necessary zero at the interior nodes of the binary tree. Using Equations (6.20) and (6.21), one can also eliminate the unknown δu's and δv's in the interior of Ω from the discrete integral constraint (8.25). Consider all adjacent regions Ω_1 and Ω_2 at the Kth level in the recursion tree, and pairwise combine their contribution to the total sum. Then (8.25) can be written

$$\sum_{\ell=1}^{2^{K-1}} \{d_1^T \delta u_1 + d_{12}^T \delta u_{12} + d_{21}^T \delta u_{21} + d_2^T \delta u_2 + t_1^T \delta v_1 + t_{12}^T \delta v_{12}$$

$$+ t_{21}^T \delta v_{21} + t_2^T \delta v_2 + (e_1 + e_2)\ \delta\lambda + s_1 + s_2\} = 0, \tag{8.26}$$

where the splitting of d, t δu, and δv is done as in Chapters 5 and 6, *cf.* Figure 5.4. Using Equations (6.21), (6.20) and (6.17), we can rewrite Equation (8.26) as

$$\sum_{\ell=1}^{2^{K-1}} \{d_1^T \delta u_1 + (d_{12}^T + d_{21}^T)(\hat{g}_{21} - \hat{D}_{21}\delta v_1 - \hat{A}_{21}\delta u_1 - \hat{D}_{22}\delta v_2 - \hat{A}_{22}\delta u_2) + d_2^T \delta u_2$$
$$+t_1^T \delta v_1 + (t_{12}^T - t_{21}^T)(\hat{g}_{12} - \hat{D}_{11}\delta v_1 - \hat{A}_{11}\delta u_1 - \hat{D}_{12}\delta v_2 - \hat{A}_{12}\delta u_2) + t_2^T \delta v_2$$
$$+(e_1 + e_2)\ \delta\lambda + s_1 + s_2\} = 0. \tag{8.27}$$

Using Equation (8.21), we have $\hat{g}_{12} = \delta\lambda\ \hat{b}_{12} + \hat{r}_{12}$, and $\hat{g}_{21} = \delta\lambda\ \hat{b}_{21} + \hat{r}_{21}$, where $\hat{g}_{12}$ and $\hat{g}_{21}$ (and so $\hat{b}_{12}$, $\hat{b}_{21}$, $\hat{r}_{12}$ and $\hat{r}_{21}$) are found from System (6.19). Equation (8.27) can be written as

$$\sum_{\ell=1}^{2^{K-1}} \{\hat{d}_1^T \delta u_1 + \hat{d}_2^T \delta u_2 + \hat{t}_1^T \delta v_1 + \hat{t}_2^T \delta v_2 + \hat{e}\ \delta\lambda + \hat{s}\} = 0, \tag{8.28}$$

where

$$\begin{aligned}
\hat{d}_1^T &= d_1^T - (d_{12}^T + d_{21}^T)\hat{A}_{21} - (t_{12}^T - t_{21}^T)\hat{A}_{11}, \\
\hat{d}_2^T &= d_2^T - (d_{12}^T + d_{21}^T)\hat{A}_{22} - (t_{12}^T - t_{21}^T)\hat{A}_{12}, \\
\hat{t}_1^T &= t_1^T - (d_{12}^T + d_{21}^T)\hat{D}_{21} - (t_{12}^T - t_{21}^T)\hat{D}_{11}, \\
\hat{t}_2^T &= t_2^T - (d_{12}^T + d_{21}^T)\hat{D}_{22} - (t_{12}^T - t_{21}^T)\hat{D}_{12}, \\
\hat{s} &= s_1 + s_2 + (d_{12}^T + d_{21}^T)\hat{r}_{21} + (t_{12}^T - t_{21}^T)\hat{r}_{12}, \\
\hat{e} &= e_1 + e_2 + (d_{12}^T + d_{21}^T)\hat{b}_{21} + (t_{12}^T - t_{21}^T)\hat{b}_{12}.
\end{aligned}$$

Note that Equation (8.28) is still of the same form as Equation (8.25), but the sum is now over half the number of regions, each of which is the union of two descendant regions in the recursive division of Ω.

The procedure is repeated recursively, and in synchrony with the eliminations that yield Equation (6.22). The final stage results in equations of the form (8.16), (8.25) (with $M = 0$), and (8.20), corresponding to the exterior boundary $\delta\Omega$. These equations can be solved for $\delta\lambda$ and for δu and δv on $\delta\Omega$. A recursive back-substitution process then gives the values of δu and δv in the interior of Ω.

8.5 Summary of the collocation algorithm

The algorithm of the finite collocation method for nonlinear PDEs with a free parameter, and with a parameter-dependent integral and boundary conditions, is very similar to that presented in Sections 5.8 and 6.6 :

- Mesh generation of 2D domain as presented in Section 5.8.
- Given current approximations to u, v, c, and λ
- Do Newton's iteration until desired relative error are achieved.
  - Loop over all elements
    - Using Equations (6.5), (6.6) and (6.9), compute the matrices Φ, L_Φ and R_Φ.
    - Solve Equation (8.15) for A and B.
    - Find r_u, r_v, r_N, f_λ, and write Equation (8.16) and its equivalent Equation (8.22) for each element.
    - Find q_u, q_λ, e and s, and compute d and $\hat{d}$ using Equation (8.18).
  - Solve the global system.
    - Construct set of Equations (8.22), (8.25), and (8.20) for the root node of the binary tree. Using the nested dissection procedure described in Section 8.4.
    - Solve the above system for $\delta\lambda$ and the unknowns δu and δv on the boundary of the domain.
    - Find the unknowns at the interior matching points of the binary tree using the recursive back-substitution, as described in Section 6.4.
  - For each finite element compute δc using Equation (8.14).
  - Update : $u \rightarrow u + \delta u$, $v \rightarrow v + \delta v$, $c \rightarrow c + \delta c$ and $\lambda \rightarrow \lambda + \delta\lambda$.
  - Compute the relative errors in the calculation of u, v and c.
- save the results.

As in previous chapters, for any given finite element, the matrix on the left hand side of Equation (8.14) is the transpose of the matrix on the left hand side of Equations (8.15) and (8.18). Thus only one LU-decomposition is required per a finite element.

8.6 Continuation

In continuous form, the pseudo-arclength equation is given by

$$\int_\Omega \big(u(x) - u_0(x)\big)\dot{u}_0(x)\, d\Omega + (\lambda - \lambda_0)\dot{\lambda} - \Delta s = 0,$$

where $(u_0(\cdot), \lambda_0)$ denotes a given solution, and $(\dot{u}_0(\cdot), \dot{\lambda}_0)$ is the direction vector of the solution branch at $(u_0(\cdot), \lambda_0)$, *i.e.*, the normalized rate of change of the solution $(u(\cdot), \lambda)$ with respect to Δs. The problem formulation in Section 8.2 includes the integral constraint (8.3). Since Ω is the unit square, with area 1, one can rewrite the continuous pseudo-arclength equations as

$$\int_\Omega \Big(\big(u(x) - u_0(x)\big)\dot{u}_0(x) + (\lambda - \lambda_0)\dot{\lambda} - \Delta s\Big)\, d\Omega = 0,$$

which is of the form Equation (8.3), with

$$q(x, u(x), \lambda) = \big(u(x) - u_0(x)\big)\dot{u}_0(x) + (\lambda - \lambda_0)\dot{\lambda}_0 - \Delta s.$$

This makes it possible to use the pseudo-arclength continuation method described in Sections 7.3 and 7.9 for computing solution families for varying λ. The discretization and the solution procedure are as described in the preceding sections. In particular, the discretized integral constraint (8.8) becomes

$$\sum_{\ell=1}^{2^M} \int_{\Omega_\ell} \Big\{\big(p(x) - p_0(x)\big)\dot{p}_0(x) + (\lambda - \lambda_0)\dot{\lambda}_0 - \Delta s\Big\} d\Omega = 0. \tag{8.29}$$

and the coefficients in the Newton equation (8.13) are now given by

$$q_u = \int_{\Omega_\ell} \begin{pmatrix} \dot{p}_0(x)\phi_1(x) \\ \vdots \\ \dot{p}_0(x)\phi_{n+m}(x) \end{pmatrix} d\Omega, \qquad q_\lambda = \int_{\Omega_\ell} \dot{\lambda}_0\, d\Omega, \tag{8.30}$$

and

$$r_q = \int_{\Omega_\ell} \Big\{\big(p(x) - p_0(x)\big)\dot{p}_0(x) + (\lambda - \lambda_0)\dot{\lambda}_0 - \Delta s\Big\} d\Omega.$$

Using a quadrature formula, Equation (8.9) can be written as

$$\sum_{\ell=1}^{2^M}\sum_{k=1}^{m}\omega_k\Big\{\big(p(z_k)-p_0(z_k)\big)\dot{p}_0(z_k)+(\lambda-\lambda_0)\dot{\lambda}_0-\Delta s\Big\}=0.$$

We also have

$$q_u=\begin{pmatrix}\sum_{k=1}^{m}\omega_k\dot{p}_0(z_k)\phi_1(z_k)\\ \vdots\\ \sum_{k=1}^{m}\omega_k\dot{p}_0(z_k)\phi_{n+m}(z_k)\end{pmatrix},\qquad q_\lambda=\sum_{k=1}^{m}\omega_k\ \dot{\lambda}_0, \tag{8.31}$$

and

$$r_q=\sum_{k=1}^{m}\omega_k\Big\{\big(p(z_k)-p_0(z_k)\big)\dot{p}_0(z_k)+(\lambda-\lambda_0)\dot{\lambda}_0-\Delta s\Big\}.$$

Above, p_0 denotes the restriction of a given piecewise polynomial solution to the ℓth element, and $\dot{p}_0$ is its derivative with respect to Δs. We can write $\dot{p}(x)=\sum_{i=1}^{n+m}\dot{c}_i\phi_i(x)$. The coefficients $\dot{c}_i$ can be determined by one extra back solve in the solution process. To see this, note that differentiating Equations (8.5), (8.6), (8.7), and (8.8) with respect to Δs gives

$$L_\Phi^T\ \dot{c}+\dot{\lambda}\ f_\lambda=0,$$

$$\dot{u}-\Phi^T\dot{c}=0,$$

$$\dot{v}-R_\Phi^T\dot{c}=0.$$

and

$$\sum_{\ell=1}^{2^M}\Big\{q_u^T\ \dot{c}+\dot{\lambda}\ q_\lambda\Big\}=1,$$

i.e., the same left hand sides as in Equations (8.10)-(8.13), with q_u and q_λ given by Equation (8.30). Thus the linear equation solution procedure of Sections 8.4 and 8.5 can be used to compute the $\dot{u}$, $\dot{v}$, and $\dot{\lambda}$. Thereafter the $\dot{c}$ can be computed from

$$\begin{pmatrix}\Phi^T\\ L_\Phi^T\end{pmatrix}\dot{c}=\begin{pmatrix}\dot{u}\\ -f_\lambda\dot{\lambda}\end{pmatrix}.$$

Chapter 9

The Collocation Method for Continuation Problems in Systems of Nonlinear Elliptic PDEs

9.1 Introduction

The discontinuous piecewise polynomial collocation method can also be used for the numerical continuation of solutions of a *system* of nonlinear elliptic PDEs. In this chapter, we follow the steps in Chapter 8, extending our work to the system of nonlinear elliptic PDEs. In fact the algorithm in this chapter is the one that has been implemented in our prototype software.

9.2 Collocation with discontinuous piecewise polynomials

Consider the following system of parameter-dependent nonlinear elliptic PDE on a unit square domain $\Omega \in R^2$

$$N_k u(x) \equiv \Delta u_k(x) + f_k(u(x), \lambda) = 0, \quad \text{for} \quad x \in \Omega, \quad k = 1, \cdots, n_f, \qquad (9.1)$$

with boundary conditions

$$u_k(x) = constant, \quad \text{for} \quad x \in \partial\Omega, \quad k = 1, \cdots, n_f, \tag{9.2}$$

and integral constraint

$$\int_\Omega q(x, u(x), \lambda) = 0, \tag{9.3}$$

where

$$x = (x_1, x_2)^T, \quad u(x) = (u_1(x), u_2(x), \cdots, u_{n_f}(x))^T.$$

In the above equations n_f is *the number of solution components*, or *the number of fields*, k is the index of solution components, Δ is the Laplace operator, Ω is an unit square in R^2, $\lambda \in R$ is the control parameter, $\delta\Omega$ is the boundary of Ω, ^{T} denotes the transpose operation, $u(x) \in R^{n_f}$, and $q \in R$, $f_k \in R$, $k = 1, \cdots, n_f$ are smooth functions. Equation (9.3) is an integral condition over the whole domain, for example, the pseudo-arclength equation (see Section 9.6).

As mentioned in the previous chapter, our collocation method and the nested dissection algorithm can be applied for problems with $x \in R^n$, the domain need not have a simple square shape and more general elements can be used. More general boundary conditions and multiple integral constraints can be treated as well. However, for simplicity of the presentation, we consider the problem corresponding to Equations (9.1) to (9.3).

The outline of our collocation method can be summarized as follows: First, a suitable mesh must be generated. For ease of calculation, programming and to be able to use the nested dissection method, as in previous chapters, the whole region is subdivided into two regions, then each region is subdivided into two subregions. This process is continued recursively until the desired mesh refinement is achieved. If we take M as the desired level of discretization of the region, or *the depth of the mesh*, the domain is divided into 2^M elements. In each element, we choose matching and collocation points. Matching points are points in common between two elements or between an element and the boundary of the domain. Collocation points, on the other hand, lie inside the element. Assume that we select m collocation points and n matching points per element. In each element, a polynomial of order $n + m$ is associated to each component (field) of the solution.

For each matching point, across the element boundary, we require the continuity of the solution components and the value of their normal derivatives. At each collocation point, the collocation equation must be satisfied. The discrete form of the integral constraint, Equation (9.3), is also imposed. If we add the discrete boundary conditions to the above equations we have a total number of

$$2^M \; n_f \; (m+n) + 1$$

equations. This nonlinear system must be solved for the $m+n$ unknown coefficients of every approximation polynomial of every solution components, and the parameter λ.

In our solution algorithm, instead of solving the above system in general form, we solve it using the nested dissection method to reduce computational cost. For a given element, we denote the matching points by $x_i,\ i = 1, \cdots, n$ and the collocation points by $z_i,\ i = 1, \cdots, m$. A polynomial $p_k(x) \in P_{n+m}$ can be used to approximate the k^{th} solution component of an element. Here, P_{n+m} is an appropriate $n+m$-dimensional polynomial space. Define $h(x) = (p_1(x), p_2(x), \cdots, p_{n_f})^T$. Then the collocation equations can be written as

$$N_k p(z_i) \equiv \Delta p_k(z_i) + f_k\Big(h(z_i), \lambda\Big) = 0, \qquad i = 1, \cdots, m, \qquad k = 1, \cdots, n_f. \tag{9.4}$$

Since for each element, we have n_f solution components, we can use the following vector basis functions to represent all of n_f polynomials together

$$\begin{array}{llll}
\varphi_1 = \begin{pmatrix} \phi_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, & \varphi_2 = \begin{pmatrix} 0 \\ \phi_1 \\ \vdots \\ 0 \end{pmatrix}, & \cdots & \varphi_{n_f} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ \phi_1 \end{pmatrix}, \\
\varphi_{n_f+1} = \begin{pmatrix} \phi_2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, & \varphi_{n_f+1} = \begin{pmatrix} 0 \\ \phi_2 \\ \vdots \\ 0 \end{pmatrix}, & \cdots & \varphi_{2n_f} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ \phi_2 \end{pmatrix}, \\
\vdots & \vdots & \vdots & \vdots \\
\varphi_{n_f(n+m-1)+1} = \begin{pmatrix} \phi_{n+m} \\ 0 \\ \vdots \\ 0 \end{pmatrix}, & \cdots & \cdots & \varphi_{n_f(n+m)} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ \phi_{n+m} \end{pmatrix}.
\end{array} \tag{9.5}$$

Here $\phi_i,\ i = 1, \cdots, n+m$, are basis functions of the polynomial space P_{n+m}.

Define a polynomial vector $h(x)$ as

$$h(x) = \sum_{i=1}^{n_f(n+m)} c_i \varphi_i(x), \tag{9.6}$$

or

$$h(x) = \sum_{k=1}^{n_f} h_k(x),$$

where

$$h_k(x) = \sum_{i=1}^{n+m} c_j \varphi_j(x), \quad \text{and} \quad j = (i-1)n_f + k.$$

We can write

$$h_1 = \begin{pmatrix} p_1 = \sum_{i=1}^{n+m} c_j \phi_j(x) \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \qquad j = (i-1)n_f + 1,$$

$$h_2 = \begin{pmatrix} 0 \\ p_2 = \sum_{i=1}^{n+m} c_j \phi_j(x) \\ \vdots \\ 0 \end{pmatrix}, \qquad j = (i-1)n_f + 2,$$

$$\vdots$$

$$h_{n_f} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ p_{n_f} = \sum_{i=1}^{n+m} c_j \phi_j(x) \end{pmatrix}, \qquad j = (i-1)n_f + n_f.$$

Then the collocation equation can be written as

$$N_k h(z_i) \equiv \sum_{j=1}^{n+m} c_{(j-1)n_f+k} \Delta \varphi_{(j-1)n_f+k}(z_i) + f_k\Big(\sum_{j=1}^{n_f(n+m)} c_j \varphi_j(z_i), \lambda\Big) = 0,$$

$$i = 1, \cdots, m, \quad k = 1, \cdots, n_f. \tag{9.7}$$

We can impose the continuity conditions by associating the unique vector variables $\overline{u}_i$ and $\overline{v}_i$ to each matching point x_i. Define

$$h(x_i) = \overline{u}_i,$$

and

$$\nabla h(x_i)\eta_i = \begin{pmatrix} \nabla p_1(x_i)^T\eta_i \\ \nabla p_2(x_i)^T\eta_i \\ \vdots \\ \nabla p_{n_f}(x_i)^T\eta_i \end{pmatrix} = \overline{v}_i,$$

where η_i is the outward normal vector to the boundary at the matching point x_i. Take

$$\widehat{u} = (\overline{u}_1^T, \cdots, \overline{u}_n^T)^T,$$

$$\widehat{v} = (\overline{v}_1^T, \cdots, \overline{v}_n^T)^T,$$

$$\Phi = \begin{pmatrix} \varphi_1^T(x_1) & \cdots & \varphi_1^T(x_n) \\ \vdots & & \vdots \\ \varphi_{n_f(n+m)}^T(x_1) & \cdots & \varphi_{n_f(n+m)}^T(x_n) \end{pmatrix}, \tag{9.8}$$

and

$$R_\Phi = \begin{pmatrix} \nabla\varphi_1(x_1)^T\eta_1 & \cdots & \nabla\varphi_1(x_n)^T\eta_n \\ \vdots & & \vdots \\ \nabla\varphi_{n_f(n+m)}(x_1)^T\eta_1 & \cdots & \nabla\varphi_{n_f(n+m)}(x_n)^T\eta_n \end{pmatrix}, \tag{9.9}$$

where

$$\nabla\varphi_i(x)^T\eta_l = \begin{pmatrix} 0 \\ \vdots \\ \nabla\phi_j(x) \\ \vdots \\ 0 \end{pmatrix}^T \eta_l = (\,0 \quad \cdots \quad \nabla\phi_j(x)^T\eta_l \quad \cdots \quad 0\,),$$

here $j = i \ mod \ n_f$, where *mod* is the modulus operator. If $j = 0$ then we take $j = n_f$. Note that in the $\nabla\varphi_i$ vector only the line j has a nonzero value, which is equal to $\nabla\phi_j(x)$. We can write

$$\widehat{u} - \Phi^T\widehat{c} = 0, \tag{9.10}$$

$$\widehat{v} - R_\Phi^T\widehat{c} = 0. \tag{9.11}$$

Above

$$\widehat{c} = (c_1, c_2, \cdots, c_{n_f(n+m)})^T.$$

The integral constraint (9.3) can be discretized over the domain, as in Section 8.2, and it can be written as

$$\sum_{\ell=1}^{2^M} \int_{\Omega_\ell} q\big(x, h(x), \lambda\big) d\Omega_\ell = 0, \tag{9.12}$$

where Ω_ℓ is the ℓth finite element of domain Ω.

Equations (9.7), (9.10), (9.11), and (9.12) with the discrete boundary conditions must be solved for

- $\widehat{c} \in R^{n_f(n+m)}$ of each element,
- $\overline{u}_i$ and $\overline{v}_i$ of each matching point, and
- the parameter λ.

9.3 Newton's method

Equations (9.7), (9.10), (9.11), and (9.12) together with the discrete boundary conditions constitute the discretization. To solve these equations, we use Newton's method. Omitting iteration indices, we can write

$$L_\Phi^T \, \delta\widehat{c} + \delta\lambda \, f_\lambda = -r_N, \tag{9.13}$$

$$\delta\widehat{u} - \Phi^T \delta\widehat{c} = -r_u, \tag{9.14}$$

$$\delta\widehat{v} - R_\Phi^T \delta\widehat{c} = -r_v. \tag{9.15}$$

and

$$\sum_{\ell=1}^{2^M} \Big\{ q_u^T \, \delta\widehat{c} + \delta\lambda \, q_\lambda + r_q \Big\} = 0. \tag{9.16}$$

where

$$\begin{aligned}
\delta\widehat{c}^T &= (\delta c_1, \cdots, \delta c_{n_f(n+m)}), \\
\delta\widehat{u}^T &= (\delta\overline{u}_1^T, \cdots, \delta\overline{u}_n^T), \\
\delta\widehat{v}^T &= (\delta\overline{v}_1^T, \cdots, \delta\overline{v}_n^T), \\
\delta\overline{u}^T &= (\delta u_1, \cdots, \delta u_{n_f}), \\
\delta\overline{v}^T &= (\delta v_1, \cdots, \delta v_{n_f}), \\
r_u &= \widehat{u} - \Phi^T \widehat{c}, \\
r_v &= \widehat{v} - R_\Phi^T \widehat{c}, \\
r_N^T &= (N_1 h(z_1), \ \cdots, \ N_{n_f} h(z_1), \ N_1 h(z_2), \ \cdots, \ N_{n_f} h(z_m)),
\end{aligned}$$

and

$$L_\Phi^T = \begin{pmatrix} L\big[h(z_1)\big]\varphi_1(z_1) & L\big[h(z_1)\big]\varphi_2(z_1) & \cdots & L\big[h(z_1)\big]\varphi_{n_f(n+m)}(z_1) \\ L\big[h(z_2)\big]\varphi_1(z_2) & L\big[h(z_2)\big]\varphi_2(z_2) & \cdots & L\big[h(z_2)\big]\varphi_{n_f(n+m)}(z_2) \\ \vdots & \vdots & \vdots\vdots\vdots & \vdots \\ L\big[h(z_m)\big]\varphi_1(z_m) & L\big[h(z_m)\big]\varphi_2(z_m) & \cdots & L\big[h(z_m)\big]\varphi_{n_f(n+m)}(z_m) \end{pmatrix}, \quad (9.17)$$

$$f_\lambda = \begin{pmatrix} D_2 f_1\big(h(z_1),\lambda\big) \\ \vdots \\ D_2 f_{n_f}\big(h(z_1),\lambda\big) \\ D_2 f_1\big(h(z_2),\lambda\big) \\ \vdots \\ D_2 f_{n_f}\big(h(z_m),\lambda\big) \end{pmatrix}.$$

Above L is the linearization of N, *i.e.*, $L[h(z)]\varphi(z)$ is the linearization of N about h acting on φ at z. More precisely,

$$L[h(z)]\varphi(z) = L[h(z)] \begin{pmatrix} \phi_1(z) \\ \vdots \\ \phi_{n_f}(z) \end{pmatrix} = \begin{pmatrix} \Delta\phi_1(z) + D_1 f_1\big(h(z),\lambda\big)\phi_1(z) \\ \vdots \\ \Delta\phi_{n_f}(z) + D_1 f_{n_f}\big(h(z),\lambda\big)\phi_{n_f}(z) \end{pmatrix}.$$

Further we have

$$q_u = \begin{pmatrix} \int_{\Omega_\ell} D_2 q\big(x,h(x),\lambda\big)\phi_1(x)d\Omega \\ \vdots \\ \int_{\Omega_\ell} D_2 q\big(x,h(x),\lambda\big)\phi_{n_f(n+m)}(x)d\Omega \end{pmatrix},$$

$$q_\lambda = \int_{\Omega_\ell} D_3 q\big(x,h(x),\lambda\big)d\Omega,$$

$$r_q = \int_{\Omega_\ell} q\big(x,h(x),\lambda\big)d\Omega.$$

The reminder of this section is very similar to Section 8.3. Equations (9.13) and (9.14) can be written

$$\begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix} \delta\widehat{c} = \begin{pmatrix} \delta\widehat{u} + r_u \\ -r_N - f_\lambda \delta\lambda \end{pmatrix}. \quad (9.18)$$

Equation (9.18) can be used to eliminate $\delta\widehat{c}$ from Equation (9.15)

$$\delta\widehat{v} = R_\Phi^T \begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix}^{-1} \begin{pmatrix} \delta\widehat{u} + r_u \\ -r_N - f_\lambda \delta\lambda \end{pmatrix} - r_v. \quad (9.19)$$

Define A and B as

$$(\,\Phi \mid L_\Phi\,) \begin{pmatrix} A^T \\ B^T \end{pmatrix} = R_\Phi. \quad (9.20)$$

Then Equation (9.19) can be rewritten as

$$\delta\hat{v} = \begin{pmatrix} A & B \end{pmatrix} \begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix} \begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix}^{-1} \begin{pmatrix} \delta\hat{u} + r_u \\ -r_N - f_\lambda \delta\lambda \end{pmatrix} - r_v,$$

or

$$\delta\hat{v} = A\delta\hat{u} - \delta\lambda B\ f_\lambda - Br_N - r_v + Ar_u.$$

If we define

$$r = -Br_N - r_v + Ar_u, \qquad \text{and} \quad b = -Bf_\lambda,$$

we can write

$$\delta\hat{v} = A\delta\hat{u} + \delta\lambda\ b + r. \tag{9.21}$$

This equation is of the form of Equation (8.16).

If we use Equation (9.18) in Equation (9.16), we have

$$\sum_{\ell=1}^{2^M} \left\{ q_u^T \begin{pmatrix} \Phi^T \\ L_\Phi^T \end{pmatrix}^{-1} \begin{pmatrix} \delta\hat{u} + r_u \\ -r_N - f_\lambda \delta\lambda \end{pmatrix} + \delta\lambda\ q_\lambda + r_q \right\} = 0. \tag{9.22}$$

Let $d \in R^{n_f n}$, $\hat{d} \in R^{n_f m}$ be the solution of

$$\begin{pmatrix} \Phi & | & L_\Phi \end{pmatrix} \begin{pmatrix} d \\ \hat{d} \end{pmatrix} = q_u. \tag{9.23}$$

Then Equation (9.22) can be written

$$\sum_{\ell=1}^{2^M} \left\{ d^T(\delta\hat{v} + r_u) + \hat{d}^T(-r_N - f_\lambda \delta\lambda) + \delta\lambda\ q_\lambda + r_q \right\} = 0,$$

which is of the form of Equation (8.19)

$$\sum_{\ell=1}^{2^M} \{ d^T \delta\hat{u} + e\ \delta\lambda + s \} = 0, \tag{9.24}$$

if we take

$$e = q_\lambda + -\hat{d}^T f_\lambda,$$

$$s = r_q + d^T r_u - \hat{d}^T r_N.$$

As in Section 8.3, the Newton equations for the boundary conditions can be written as

$$\delta u = -u, \tag{9.25}$$

where δu and u are n_f component vectors. This allows treating more general boundary conditions.

9.4 Nested dissection

The nested dissection method described in Section 8.4 can be applied to the global set of Equations (9.21) and (9.24). The algorithm consists of backward recursive elimination of the unknowns $\delta\bar{u}$ and $\delta\bar{v}$ on boundaries separating adjacent regions. Having used this procedure, we find the equations of the form Equation (9.21) corresponding to the matching points on the exterior boundary $\delta\Omega$. The discrete boundary conditions can be used to determine $\delta\lambda$ and the values of $\delta\bar{u}$, $\delta\bar{v}$ at the points x_i on $\delta\Omega$. Thereafter a recursive back-substitution gives the values of $\delta\bar{u}$ (and hence $\delta\bar{v}$) on each interior boundary point.

If we replace δu by $\delta\hat{u}$ and δv by $\delta\hat{v}$, we can use the procedure described in Section 8.4. Consequently, we do not repeat it here. Note that $\delta\bar{u}$ and $\delta\bar{v}$ are vectors of n_f components, for each matching point.

9.5 Summary of the collocation algorithm

The algorithm of the finite element collocation method for systems of nonlinear PDEs with a parameter-dependent integral boundary condition is very similar to the algorithm in Section (8.5):

- Mesh generation of 2D domain as presented in Section 5.8.
- Given current approximations to $\bar{u}$, $\bar{v}$, $\hat{c}$ and λ. ($\bar{u}$ and $\bar{v}$ are vectors of n_f elements.)
- Do Newton's iterations until the desired relative errors are achieved.
  - Loop over all elements
    * Compute Φ, L_Φ and R_Φ matrices using Equations (9.8), (9.9) and (9.17).
    * Solve Equation (9.20) to find A and B.
    * Find r_u, r_v, r_N, f_λ as described in Section 9.3, and write Equation (9.21) and its equivalent Equation (8.22) for each element.
    * Find q_u, q_λ, e and s, and compute d and $\hat{d}$ using Equation (9.23).

- Solve the global system.
  - * Set up Equations (8.22), (8.25), (9.25) for the root node of the binary tree.
  - * Use the nested dissection procedure described in Sections 8.4 and 9.4.
  - * Solve the above system for $\delta\lambda$ and the unknowns $\delta\widehat{u}$, $\delta\widehat{v}$ on the boundary of the domain.
  - * Find the unknowns at the interior matching points of the binary tree using the recursive back-substitution, as described in Section 6.4.
- For each finite element compute $\delta\widehat{c}$ using (9.18).
- Update : $\bar{u} \rightarrow \bar{u} + \delta\bar{u}$, $\bar{v} \rightarrow \bar{v} + \delta\bar{v}$, $\widehat{c} \rightarrow \widehat{c} + \delta\widehat{c}$ and $\lambda \rightarrow \lambda + \delta\lambda$.
- Calculate the relative errors in the calculation of $\bar{u}$, $\bar{v}$ and $\widehat{c}$.

- save the results.

For any finite element, the matrix on the left hand side of (9.18) is the transpose of the matrix on the left hand side of (9.20) and (9.23). Thus only one *LU*-decomposition is required per finite element.

9.6 Continuation

The pseudo-arclength continuation method presented in Sections 7.3 and 7.9 can also be used for the path following problems in systems of nonlinear elliptic PDEs with control parameters. The nonlinear system (9.1) can be presented in the following compact form

$$G(\bar{u}(x), \lambda) = 0, \qquad \bar{u}, G \in R^{n_f}, \ \lambda \in R, \tag{9.26}$$

here $\bar{u}(x)$ is a function vector of n_f components and λ is a system control parameter. The pseudo-arclength continuation equations can be formulated as

$$G(\bar{u}(s), \lambda(s)) = 0, \tag{9.27}$$

$$\int_{\Omega} \big(\bar{u}(x) - \bar{u_0}(x)\big) * \dot{\bar{u}}_0(x)\, d\Omega + (\lambda - \lambda_0)\dot{\lambda} - \Delta s = 0, \tag{9.28}$$

where s is a parameterisation, $*$ denotes dot product, $(\bar{u_0}(\cdot), \lambda_0)$ denotes a given solution, and $(\dot{\bar{u}}_0(\cdot), \dot{\lambda}_0)$ is the direction vector of the solution branch at $(\bar{u_0}(\cdot), \lambda_0)$.

Using $h(x)$, defined in Equation (9.6), as an approximation to $\bar{u}$, Equation (9.28) can be written as

$$\int_{\Omega}(h_1(x) - h_0(x)) * \dot{h}_0(x)d\Omega + (\lambda_1 - \lambda_0)\dot{\lambda}_0 - \Delta s = 0. \tag{9.29}$$

To find the new solution point (h_1, λ_1) from the known solution point (h_0, λ_0), the following equations must be solved

$$G(h_1, \lambda_1) = 0,$$

$$\int_{\Omega}(h_1(x) - h_0(x)) * \dot{h}_0(x)d\Omega + (\lambda_1 - \lambda_0)\dot{\lambda}_0 - \Delta s = 0.$$

As Ω is a unit square, with area 1, one can rewrite the pseudo-arclength equation (9.29) as

$$\int_{\Omega}\{(h_1(x) - h_0(x)) * \dot{h}_0(x) + (\lambda_1 - \lambda_0)\dot{\lambda}_0 - \Delta s\}d\Omega = 0. \tag{9.30}$$

This equation is of the form of Equation (9.3), where

$$q(x, u(x), \lambda) = \Big(h(x) - h_0(x)\Big) * \dot{h}_0(x) + (\lambda - \lambda_0)\dot{\lambda}_0 - \Delta s.$$

We can apply the pseudo-arclength equation as an integral constraint and we can use the discretization method presented in Section 9.2 and 9.3. The procedure is almost the same as in Section 8.6. Our nonlinear system comprises Equations (9.7), (9.10), (9.11), the boundary conditions, and the discretized pseudo-arclength integral constraint. The later equation can be written as

$$\sum_{\ell=1}^{2^M}\int_{\Omega_\ell}\Big\{\Big(h(x) - h_0(x)\Big) * \dot{h}_0(x) + (\lambda - \lambda_0)\dot{\lambda}_0 - \Delta s\Big\}d\Omega = 0.$$

The coefficients of Equation (9.16) are

$$q_u = \begin{pmatrix} \int_{\Omega_\ell}\dot{h}_0(x)\phi_1(x)d\Omega \\ \vdots \\ \int_{\Omega_\ell}\dot{h}_0(x)\phi_{n_f(n+m)}(x)d\Omega \end{pmatrix}, \qquad q_\lambda = \int_{\Omega_\ell}\dot{\lambda}_0\, d\Omega, \tag{9.31}$$

and

$$r_q = \int_{\Omega_\ell}\Big\{\Big(h(x) - h_0(x)\Big) * \dot{h}_0(z_k) + (\lambda - \lambda_0)\dot{\lambda}_0 - \Delta s\Big\}d\Omega.$$

We remark that h_0 denotes the restriction of a given piecewise polynomial solution to the ℓth element, and $\dot{h}_0$ is its derivative with respect to Δs. We can write

$$\dot{h}(x) = \sum_{i=1}^{n_f(n+m)}\dot{c}_i\varphi_i(x).$$

The next direction vector $(\dot{h}_1, \dot{\lambda}_1)$ is defined by

$$\begin{aligned} G_h^1 \dot{h}_1 + G_\lambda^1 \dot{\lambda}_1 &= 0, \\ \int_\Omega \dot{h}_1 * \dot{h}_0 d\Omega + \dot{\lambda}_0 \dot{\lambda}_1 &= 1. \qquad \text{(normalisation)} \end{aligned}$$

As mentioned in Section 8.6, the coefficients $\dot{c}_i$, and therefore the vector $(\dot{h}_1, \dot{\lambda}_1)$, can be determined by one extra back solve in the solution process. We must also rescale the direction vector, such that

$$\|\dot{h}(s)\|^2 + \dot{\lambda}(s)^2 = 1,$$

where

$$\|\dot{h}(s)\|^2 = \int_\Omega \dot{h}(x) * \dot{h}(x) d\Omega.$$

Chapter 10

Implementation

10.1 Implementation considerations

There is no doubt that for the modelling and the implementation of a complex software package, such as one for bifurcation analysis, a good modelling technique is essential. *Object-oriented modelling* of the bifurcation analysis of systems of nonlinear PDEs using discontinuous piecewise polynomial collocation, gives us the opportunity to present the collocation and the continuation method clear and in a convenient way. Such modelling is a useful tool to present the geometrical components of the collocation model as well as the mathematical relations between these components. Object-oriented modelling shows how different components communicate to construct the collocation system, and how this system is used in pseudo-arclength continuation. It shows visually the continuation algorithm, and how folds and branch points can be detected. An object-oriented model is not only useful for software construction, but it also is a valuable tool for understanding a complicated procedure such as path following.

There are many factors in a project's success, and among them having a rigorous modelling language standard is fundamental. A good modelling language must include: *model elements* (fundamental modelling concepts and semantics), *notation* (visual presentation of model elements) and *guidelines* (idioms of usage within the context). In this thesis, we have selected the *UML* (Unified Modelling Language) notation as our object-oriented modelling language. The Rational Rose software is a powerful tool that can be used for the object-oriented modelling with UML [109]. Unfortunately, the Rational Rose software package is a commercial software package, as a result, we have used a student version of this software (version 4.0), and also the open source ArgoUML [107] which is valuable software as well.

We have used the UML notation, based on OMG Unified Modelling Language Specification Version 1.5 [110], in the presentation of our prototype software structure. We present our object-oriented model using

- *Class diagrams* for the presentation of the static structure of our model in terms of *classes* and their *associations* (relationships),
- *Sequence diagrams* for the temporal representation of the activity of the *objects* of our model, and the interactions among these objects,
- *Collaboration diagrams* for the spatial representation of the collaborating objects, their links, and their interactions,
- *Object diagrams* for the presentation of the objects of our model and their relationships,
- *Statechart diagrams* for the presentation of the behavior of the classes in terms of states,
- *Activity diagrams* for the presentation of the behavior of a given operation as a set of actions,
- *Use case diagrams* for the presentation of functions of our system from the user's view point,
- *Component diagrams* for the presentation of the physical components of our prototype software.

- *Deployment diagrams* for the presentation of the deployment of the components of software on the given hardware.

Using these diagrams make understanding the structure of the software and the behavior of our system clear.

10.2 Choice of programming language

We need an object-oriented programming language advanced enough to be able to handle our data structure. We need to generate and to manipulate the objects dynamically. Such a language must enable us to carry out the basic mathematical operations efficiently, having acceptable input/output functionalities, and let us to build a portable program that can be executed in different operating systems. The produced software must also be compatible with other popular software packages available in the domain of scientific computation, especially ones use in the numerical bifurcation analysis, such as AUTO. These considerations led us to select the C++ language.

Besides the object-oriented advantages of C++, one of the principal forces of this language is its capacity to handle pointers (heritage of C). So we can write efficient low level subroutines. It allows to allocate (and to release) memory dynamically. This will enable us to manage very naturally the lists, the tables, the trees, the containers and the objects which successively appear, evolve and then possibly disappear in the course of execution of program.

Another important point is the link to the available linear algebra software packages, such as LAPACK++, to perform mathematical operations or the use of the standard classes to execute certain tree structure operations that may improve the performance of our software. Utilization and links to these packages are possible easily when we use C++ as the programming language.

The C++ language allows freedoms which would make most compilers quiver. It authorizes some powerful instructions that must be used carefully. Moreover, with an aim of generating the most effective possible code, the C and C++ languages have certain syntactic turning which, in addition to being of a delicate use, make the source

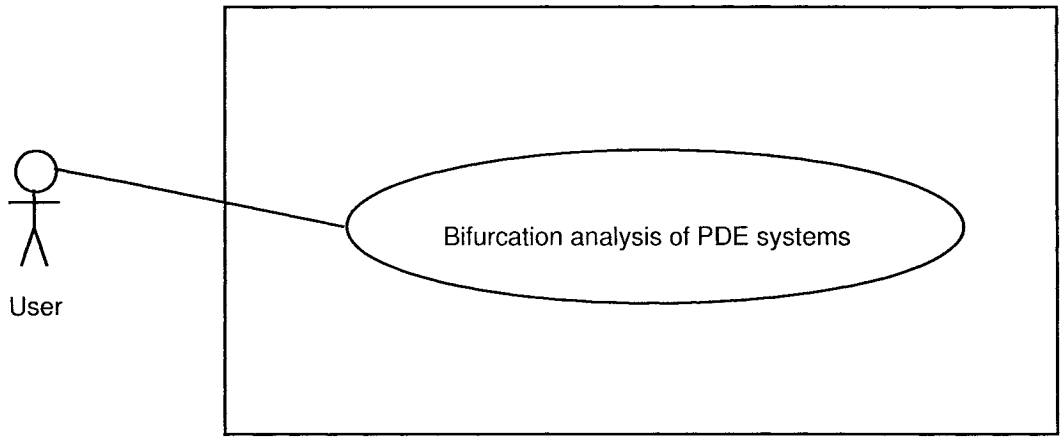


Figure 10.1: Use case diagram of the entire system

code less readable and thus more difficult to understand. The places where one uses these forms must be well documented.

10.3 Use Case model of the system

A *use case model* presents relationships and message exchanges between users of the program which are called *actors* and the functionalities of the program which are called *use cases*. In this model, the functionalities of the system are represented as appear for external users of the system. A use case diagram is a graph of actors, a set of use cases enclosed by the system boundary, communication (participation) associations between the actors and the use cases, and generalizations among the use cases. Each use case, shown as an "ellipse" containing the name of the use case, consider as a unit of functionality provided by a system. An actor is a role of object (or objects) outside of the system that is in interaction with the system, *i.e.* the use cases of the system. The standard stereotype icon for an actor is the "stick man" figure with the name of actor below it.

In our system, a system for bifurcation analysis of PDE systems, the whole system can be considered as a use case, which is in the interaction with a user, who can be a person or even a program. This overall utilization of the system is shown in the use case diagram of Figure 10.1. The link between the user and the use case represents the messages exchange between them.

If we want to look at the system in more detail, we can break the use case

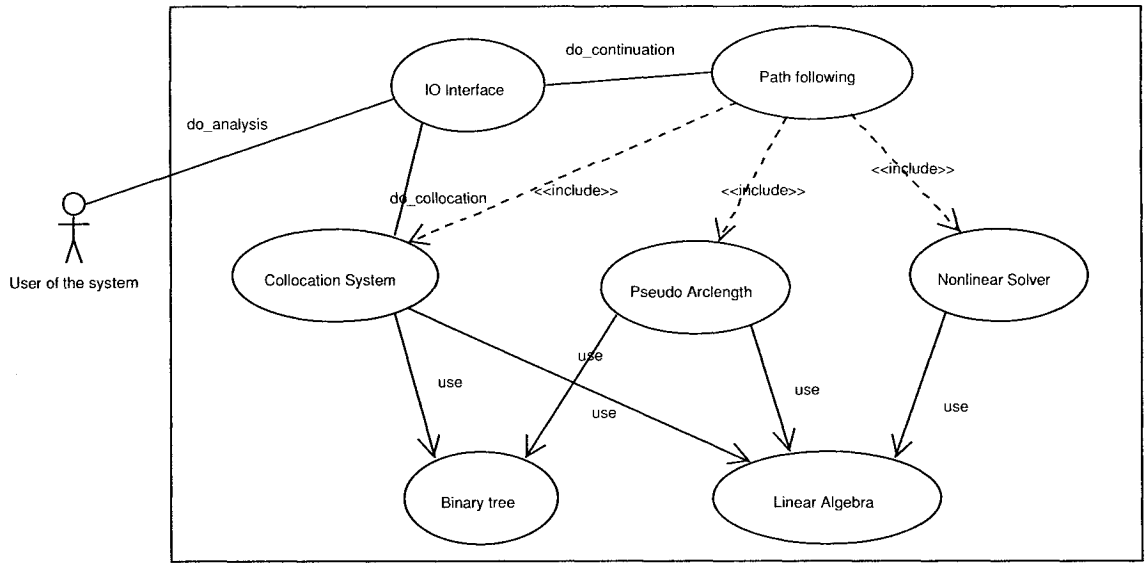


Figure 10.2: Detailed use case diagram of the system

"Bifurcation analysis of PDE systems" into several use cases then, the interactions between an actor and these use cases can be shown. The detail use case diagram is presented in Figure 10.2 As shown in this diagram, the user communicates with "IO Interface" (input output interface) use case asking for an analysis. "IO Interface" provides some functionalities to the user to enable him to inter the domain specifications, the system of elliptic PDEs, the boundary conditions and also the mesh specification. The result of calculations is also sent to this use case. Therefore, "IO Interface" is also responsible to prepare an appropriate output, and to communicate it to the user via data sheets or different diagrams. The "IO Interface" use case after collecting the input data send a message to "Path following" use case asking for the execution of the continuation algorithm. The "Path following" use case must construct the collocation system, add pseudo-arclength equation to it, and solve the nonlinear system. These three steps can be shown as three use cases, which include in "Path following" use case. As a result, "Collocation System", "Pseudo-Arclength", and "Nonlinear Solver" use cases are added to the diagram and there is an include relation between "Path following" and each of these three use cases. "Collocation System", "Pseudo-Arclength" and "Nonlinear Solver" use cases use "Linear Algebra" use case in their calculation but "Linear Algebra" use case cannot be considered as part of these use cases.

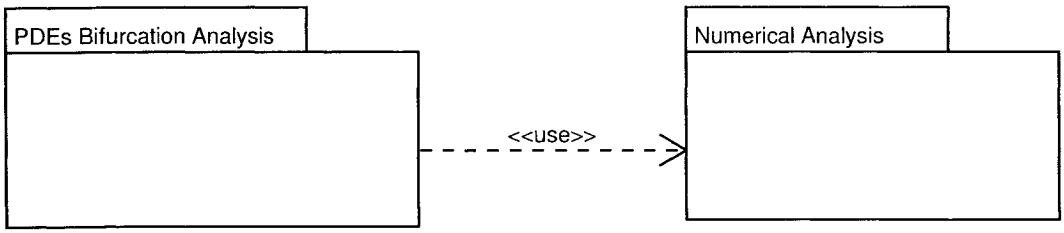


Figure 10.3: Package diagram of the system

The link between "IO Interface" use case and "Collocation System" use case shows a situation when the user only wants to solve a PDE system without doing a bifurcation analysis. In this case, "IO Interface" only communicates with "Collocation System". "Collocation System" and "Pseudo-Arclength" use cases need geometrical and mesh information of the domain. Such information is store in "Binary tree" use case.

10.4 Static structure of the model

10.4.1 Packages

Model elements such as objects and classes can be organized using *packages*. In our model, we can use a single package to keep our developed model elements, which are called "PDEs Bifurcation Analysis". In case of using external numerical analysis software packages, for example, linear and nonlinear solvers, we can put them in a "Numerical Analysis" package as shown in Figure 10.3. As "PDEs Bifurcation Analysis" uses the "Numerical Analysis" package, there is a "use" relationship between these two packages.

10.4.2 Classes

A class symbolizes a set of objects with similar data structure, behavior, and relationships to other elements. A class can represent a concept within the system. For each class a list of *attributes* and a list of *operations* (*methods*) must be declared. Class diagrams show the static structure of the system, *i.e.* the elements that define for the overall life time of a program, such as classes, their internal structure, and their relationships to each other.

The following classes are declared in our object-oriented model:

- "User Interface": This class is the realization of "IO Interface" use case. It creates an environment for a user to inter the system of PDEs, the domain specification, the boundary conditions and also the mesh specification. This class must also communicate the result of calculations to the user.

- "Element": This class generates the elementary system of equations for each element, as describes in the previous chapters. The objects created from this class form the leaves of the binary tree data structure of the problem.

- "Region": In our binary tree data structure, a region is an interior node of the tree. Geometrically, each region is a domain constructed from the union of two subregions originally formed from the division of it, *i.e.*, in the mesh generation procedure. Using the class "Region", we can create the binary tree structure. In this class, we must have the methods for assembling the equations of its subregions to find the system of equations of the region, using the nested dissection method. Therefore, when a region is the root of the binary tree, we can have the linearized system of equations for the whole domain. As elements are the smallest subregions of the domain, they can also be considered as regions. As a matter of fact, the class "element" is a specialization of the class "region" and there is a generalization relationship between them, as shown in Figure 10.4. Note, we have only shown a few methods of these classes in this figure.

- "Node": Class Node models a point of the domain. This point could be a mesh point, a matching point or a collocation point. Using the class Node, we can create geometric points (nodes), and for each point, we can store its coordinates, its type and other properties associated to a point.

- "Bifurcation Analyzer": This class provides attributes and operations to perform a bifurcation analysis of a PDE system. There is a link from this class to the User Interface class, see Figure 10.5. This link shows the possibility of data transfer between these two classes. As a result the PDE system defined in the User Interface class is accessible from the "Bifurcation Analyzer" class. The result of analysis can also be transferred to the User Interface class to communicate it to the user. Another link from the Bifurcation Analyzer class to

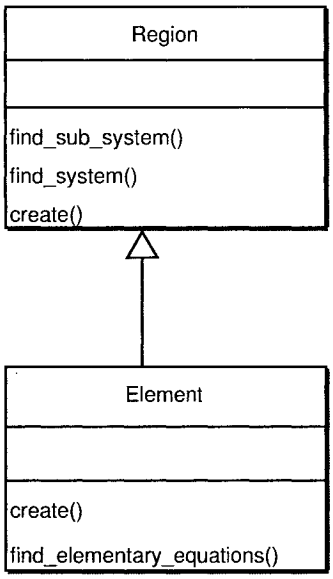


Figure 10.4: Region and element classes

the Region class permits this class to have access to the linearized system of equations of the root region.

- "Pseudo-Arclength": This class is responsible for forming the pseudo-arclength equation. The Pseudo-Arclength class has access to the methods defined in the Bifurcation Analyzer and the Element classes in the process of its calculation.

- "Linear Solver": We use this class, when we need to solve a linear system of equations. This class uses the LU decomposition method to solve the linear system, and it takes Matrix and Vector objects as the input arguments.

- "Nonlinear Solver": This class uses the Newton method for solving a nonlinear system of equations. As in each iteration of the Newton method, we need to solve a linear system, this class has access to the Linear Solver class.

- "Matrix" : This class provides attributes and methods to store and to execute different operations on a matrix structure.

- "Vector": In this class, we define different attributes and methods to store a vector and to execute different operations on it. As a vector is an special case of a matrix, there is a generalization relationship between these two classes, *i.e.*, the Vector class is a child of the Matrix class.

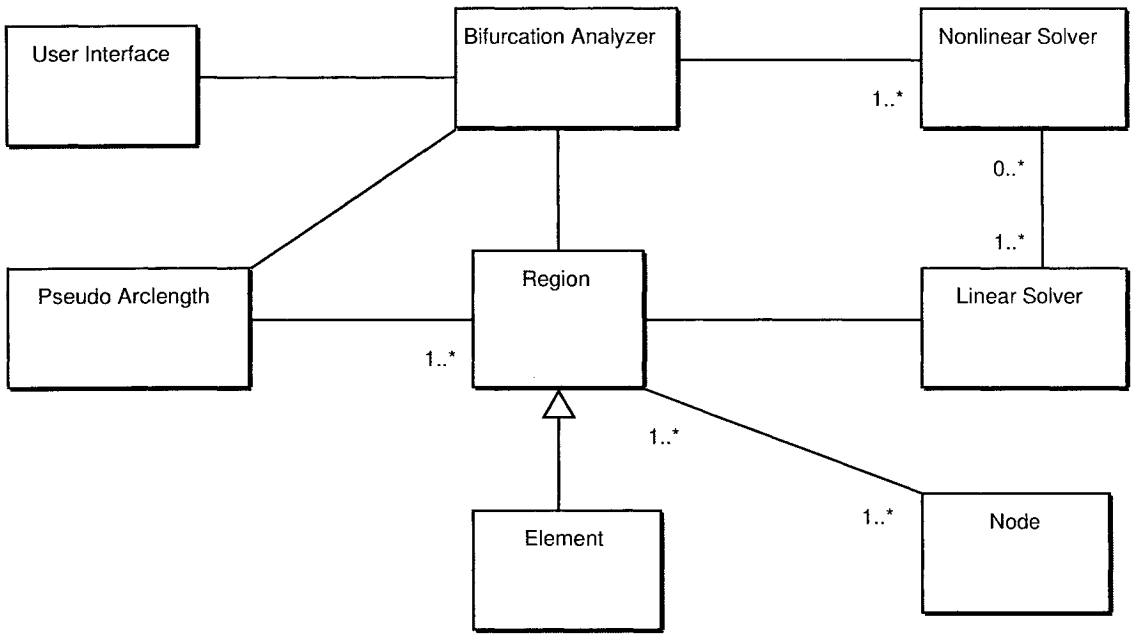


Figure 10.5: General class diagram of the system

- "SubMatrix": As in our calculation, especially in the nested dissection method, we need to work with a part of a matrix, we define this class. A SubMatrix class is defined over a Matrix. As a result, we work on a partition of the original matrix the same as we work on a real matrix. As we work on the same storage space the modification on the SubMatrix is passed to the original matrix.

The general class diagram of our object oriented model is shown in Figure 10.5. A region can associate with several nodes, and a node can also belong to several regions, as a result, as shown in the diagram, there is a $1..*$ association between the Region and the Node classes. A Bifurcation Analyzer needs to communicate with a User Interface, Regions, a Pseudo-Arclength and a Nonlinear Solver to construct the system of equations and to solve it. Therefore the links shown in Figure 10.5 exists between these classes. We consider the possibility that a Bifurcation Analyzer can use different Nonlinear Solvers, so there is the multiplicity $1..*$ in the Nonlinear Solver side of the link connecting these two classes. A Nonlinear Solver can also use different Linear Solvers, therefore, we have $1..*$ multiplicity on the Linear Solve side of the link between them. A Linear Solver can also relate to several Nonlinear Solvers but it is not necessary for a Linear Solver to relate to any Nonlinear Solver, as a result, we have $0..*$ multiplicity on the Nonlinear Solver side of the link between these classes.

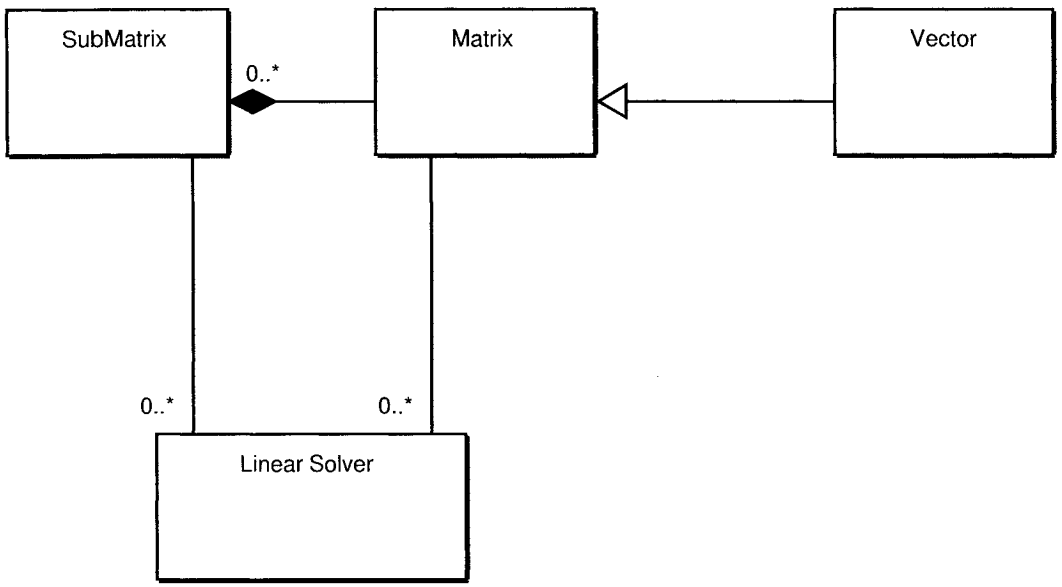


Figure 10.6: Class diagram of the linear solver

The link between Region and Linear Solver classes shows a Region has access to a Linear Solver for its internal calculation. A Pseudo-Arclenth can also communicate with Regions to collect the necessary data for its calculation.

Figure 10.6 shows relationships between SubMatrix, Matrix, Vector and Linear Solver classes. As one can see, a matrix is a component of a submatrix and a vector is a child of a Matrix. A linear solver can relate to a matrix or a submatrix but a matrix or a submatrix can be used in several linear solvers.

10.5 Interaction among objects in the bifurcation analysis

The bifurcation analysis can be implemented by sets of *objects* that exchange *messages*, *cf.* to Figure 10.7. Before examining this figure, several definitions are reviewed. A communication between two or several objects is called a message. A message contains the information that the sender transfers with the hope that an action can take place. The reception of a message is an *event*. An object represents a particular instance of a class. It has a name, and it stores values of its attributes. The object notation comes from the class notation, to show the presence of an object in a particular instance of a class; the syntax: *objectname: classname* is used. A

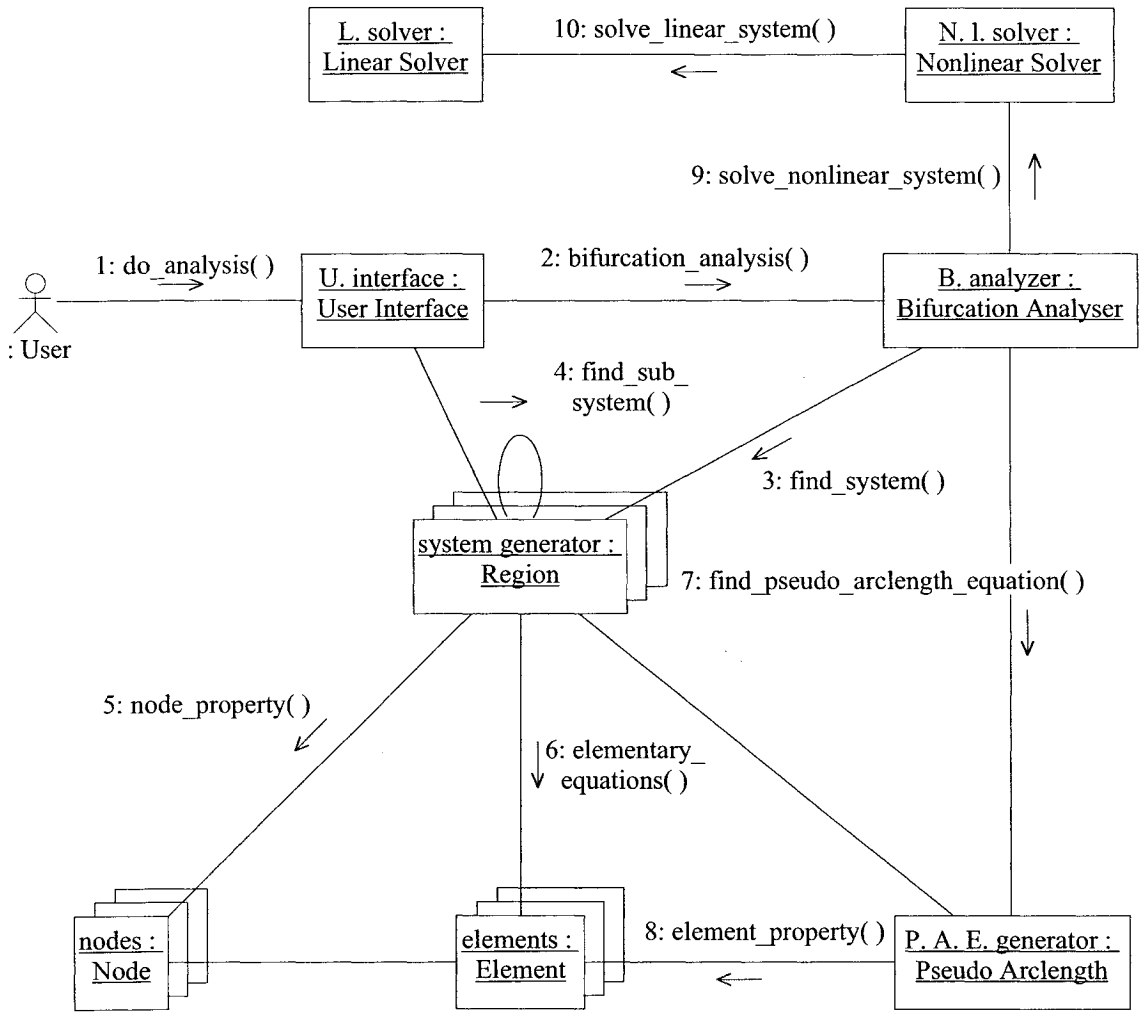


Figure 10.7: Collaboration diagram of the bifurcation analysis

multi-objects is shown as several rectangles; some rectangles are shifted slightly vertically and horizontally to present a stack of rectangles. The dynamic behavior of the message sequences exchanged among objects to perform a specific purpose, such as a bifurcation analysis, is an *interaction.* The structure of objects playing roles in an interaction and their relationships is a *collaboration.* A collaboration diagram is a graph of objects and their links with message flows attached to the links. In our case, this diagram can present the objects involve in the continuation procedure, including objects indirectly involved or accessed during the operations.

Figure 10.7 presents the collaboration diagram of the bifurcation analysis. It demonstrates the relationships between the objects, and an interaction organized

around them to perform a bifurcation analysis. As shown in this figure, the interaction is started when a user sends a message "do analysis" to the object "U. interface" of type Class User Interface. The object "U. interface" sends the message "bifurcation analysis" to the object "B. analyzer" of type Class Bifurcation Analyser. This object, to find the root node linearized system of equations at each step of the continuation method, sends messages "find system" to the objects "system generator" of type Class Region and "find pseudo-arclength equation" to the object "P. A. E. generator" of type Class Pseudo-Arclength. The object "B. analyzer", after receiving the requested equations, constructs the root node system and sends it via the message "solve nonlinear system" to the object "N. L. solver" of type Class Nonlinear Solver to find the solution. The object "N. L. solver" uses the Newton method to solve the nonlinear system and at each iteration of the Newton method sends a linear system via the message "solve linear system" to the object "L. solver" of type Class Linear Solver to find the solution.

The objects "system generator" are really the binary tree of the system. The message "find system" is sent to the root of the binary tree and the root sends the message "find sub-system" to each of its children. Each child sends the same message to its children until one level before the leaves of the tree, where each object sends the message "elementary equations" to its associated leaves of the tree. Leaves of the tree are the objects "elements" of type Class Element. The "system generator" and the "elements" objects need the mesh points, the collocation points and the matching points properties. These properties are stored in the objects "nodes" of type Class Node. Therefore, the "system generator" and the "elements" objects send the message "node property" to the "nodes" objects. The "P. A. E. generator" object also needs the elements properties, as a result, it sends the message "element property" to the "elements" objects .

In the collaboration diagram, the sequence of messages is shown using an order number written before each message. To see the explicit sequence of messages in time, we can use the sequence diagram presented in Figure 10.8. The sequence diagram shows the same interaction, but it is arranged in time sequence. It shows the objects participating in the interaction by their *lifelines* and the messages that they exchange within a collaboration to produce a bifurcation analysis. This diagram has

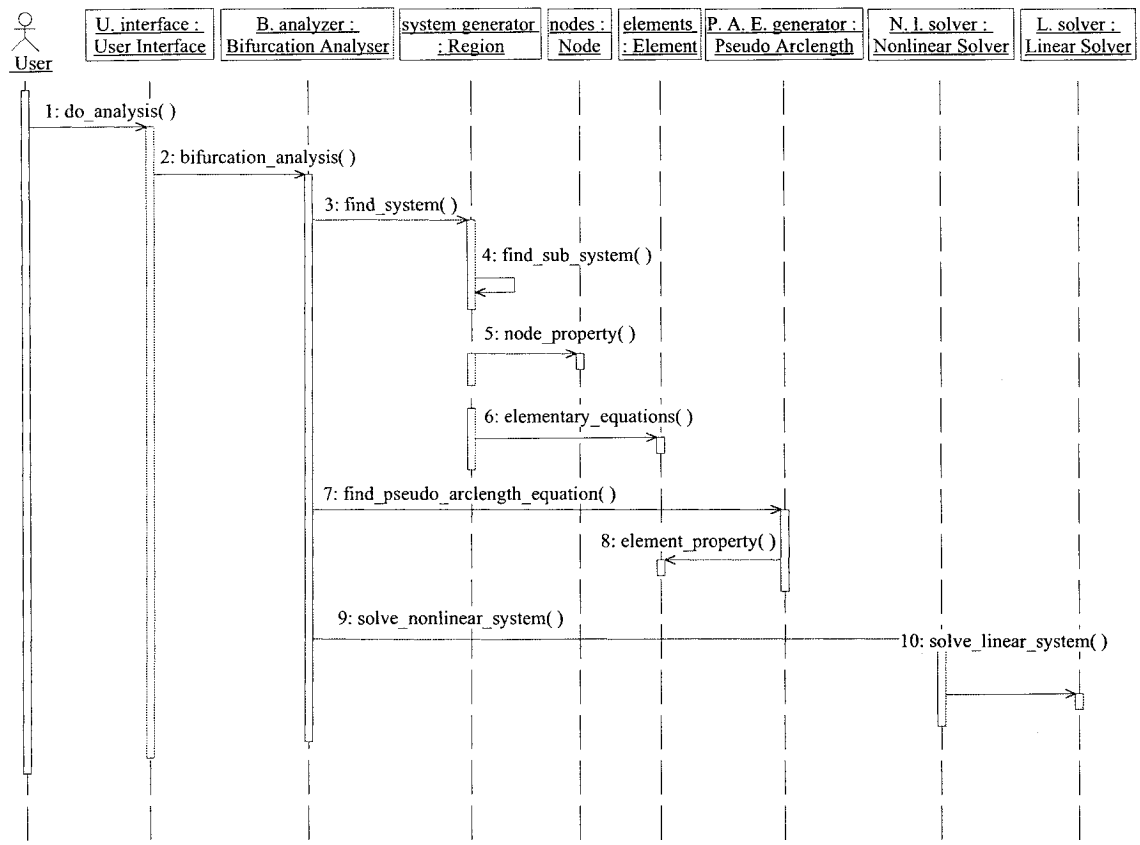


Figure 10.8: Sequence diagram of the bifurcation analysis

two dimensions: the vertical dimension represents time and the horizontal dimension represents different objects. An object role is shown as a vertical dashed line called the lifeline. The lifeline represents the existence of an object at a particular time. An *activation* shows the period during which an object is performing an action. The activation is shown as a tall thin rectangle whose top is aligned with its initiation time and whose bottom is aligned with its completion time. A message is shown as a horizontal solid arrow from the lifeline of one object to the lifeline of another object. In case of a message from an object to itself, the arrow may start and finish on the same object lifeline.

As one can see, the user activates the "U. interface" object when it sends the message "do analysis" to it. The "U. interface" activates "B. analyzer". The "B. analyzer" object activates the "system generator", the "pseudo-arclength" and the "N. L. solver" objects. In this diagram, we only present the sequential calculation of the model, therefore the "pseudo-arclength" is activated after the execution of the

"system generator". In a parallel calculation the "system generator" and the "pseudo-arclength" objects can be activated at the same time. The "N. L. solver" object must be activated after the execution of the "system generator" and the "pseudo-arclength" objects. The "system generator" objects have recursive call. They also activate the "nodes" and the "elements" objects. The "L. solver" is also activated via the "N. L. solver" object.

The above interaction diagrams present one step of the continuation procedure. Such collaboration and interaction must be repeated for every step, as we can store the result of the repeated calculations, the fist step takes more time than others.

10.6 Different states of the bifurcation analysis

As the principal job in a bifurcation analysis is done by an object of type Class Bifurcation Analysis, and this object control the others, in this section we consider the sequences of *states*, which such object could have during its life. Note, a state is a situation during the life of an object, which it satisfies some conditions, performs some actions, or waits for some events. Here, an event is a notable situation. An event may trigger a state transition in the state diagrams. In our modelling, we have used the statechart diagram to present the different states of a Bifurcation Analysis object in reply to a request of doing an analysis, refer to Figure 10.9.

In the UML, a state is represented by a state symbol, which is a rectangle with rounded corners. Arrows connecting the state symbols represent the transitions. A state symbol may have one or more compartments. Name compartment contains the name of the state. Internal transition compartment contains a list of internal actions or activities performed in response to the received events during which the object is in the state, without changing its state. An initial (pseudo) state is presented by a small solid filled circle. A final (pseudo) state is shown by a circle that enveloped a small solid filled circle (a bull's eye). The trigger for a transition between states is the occurrence of the event labelling the transition.

The Bifurcation Analysis object, when triggered, constructs the binary tree data structure of the domain, then it searches for the starting point of the continuation, which is generally a travail solution of the problem. We call this state "Initialization".

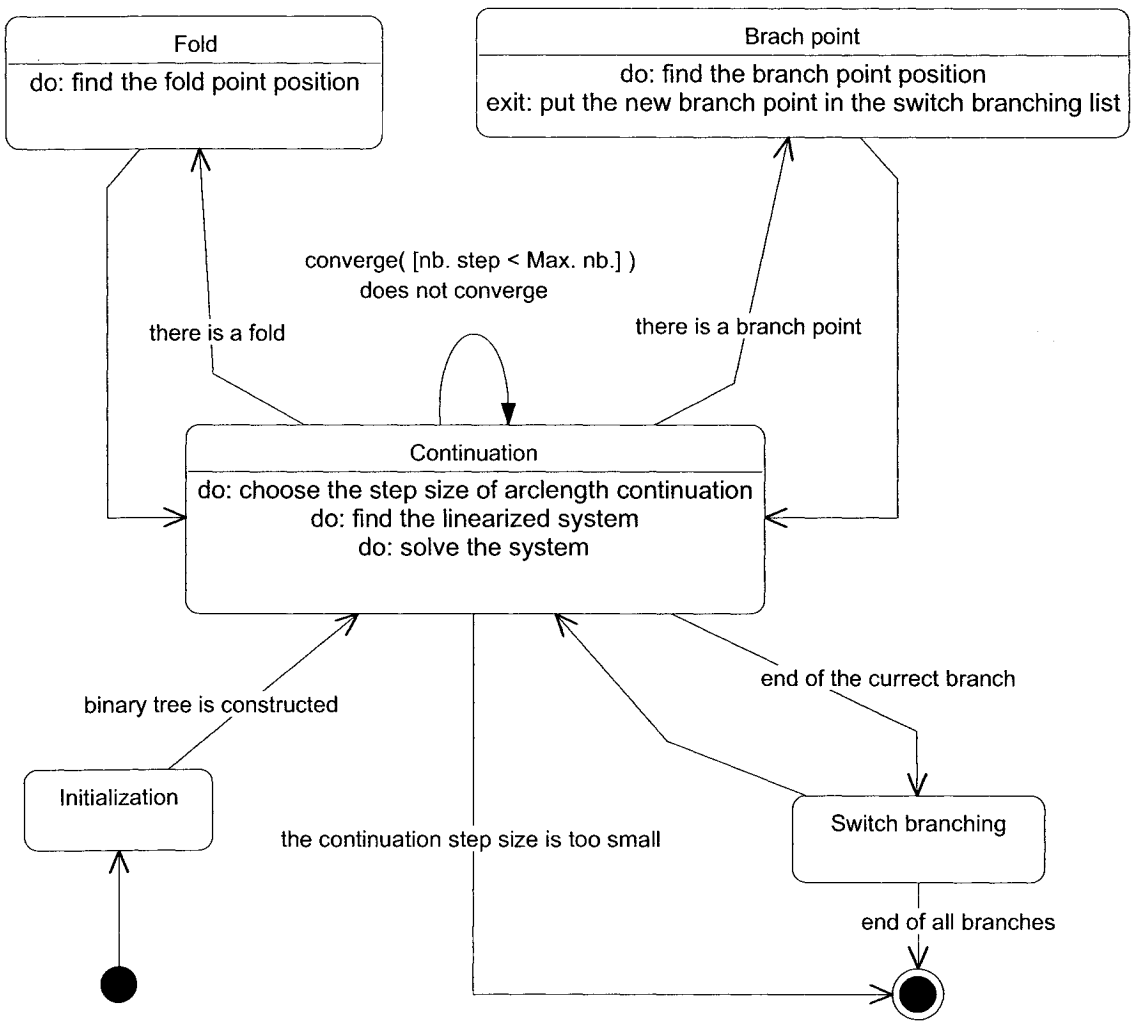


Figure 10.9: Statechart diagram of the bifurcation analysis object

After the Initialization state, the Bifurcation Analysis object goes to the "Continuation" state. In this state the following actions will be performed:

- choosing the step size of arclength continuation,
- finding the linearized system, and
- solving the system.

As we use the pseudo-arclength continuation method, first we must choose an arclength step (Δs). In the first action the size of Δs must be selected. The value of Δs depends on its initial value, the previous step size, the convergence condition, and the number of Newton's iterations of the previous step for solving the system. The second action is finding the linearized system of equations. This system has two parts. First part is formed by assembling of the elementary equations of all domain elements, which we have used the modified nested dissection method as described in the previous chapters. The seconds part is the pseudo-arclength equation, which is written for the whole domain. The third action of this state is solving the system using the Newton method. Each of the above actions can be presented in a statechart or in an activity diagram. We present the third action in an activity diagram in the next section. After solving the system the following events could be happened:

- the system does not converge,
- the system converges and the limit of continuation on this branch has not been reached (number of steps on this branch is less than the given value),
- there is a fold,
- there is a branch point,
- end of the current branch, the limit of continuation on this branch has been reached
- the continuation step size, Δs, is too small.

If the system does not converge, then the Bifurcation Analysis object must stay in its current state, and it selects a smaller continuation step size, for example it can choose $\Delta s_{i+1} = \Delta s_i/2$. Therefore, for this event, we have a self transaction. If the system converges, and the given continuation limit on this branch has not been reached, then it must also stay in its current state, and it goes one step forward on the branch. As a result, we have another self transaction event.

If there is a fold on the branch, detected by an orientation change of the direction vector, we need to find its position. Here, the Bifurcation Analysis object changes state from the "Continuation" to the "Fold". In the Fold state, it uses the Regula-Falsi or the bisection method to locate the fold point with the precision that has been requested by the user. After finding the fold point, it returns back to the "Continuation" state, and it continues to step forward from the fold point on the current branch.

If there is a branch point, detected by the sign change of the determinant of the linearized system, then the Bifurcation Analysis object goes to the "Branch point" state to find the position of this point with the desired precision. After finding the location of the branch point, it stores its coordinates, the solution of the unknowns at this point, and the direction vectors associated to this point in the list of switch branching. This information, together with each of the direction vectors associated to the branch point, represent a branch of solution that can be used in the continuation method to find all the solution branches. Thereafter, it returns back to the "Continuation" state, and it continues to step forward from the branch point on the current branch.

If the limit of the continuation on the current branch has been reached, there is no need to go further on the current solution branch, and the Bifurcation Analysis object changes state to the "Switch branching" state. In the "Switch branch" state, it searches for a new branch point from the list of switch branching. If found, it loads the new branch point information, and it returns back to the "Continuation" state to do continuation on the new branch. If the new branch point is not found on the switch branching list, it goes to the final state.

If the "too small continuation step size" event occurs, then there must be an error in the program or in the input data. Therefore, an error signal arises, and the

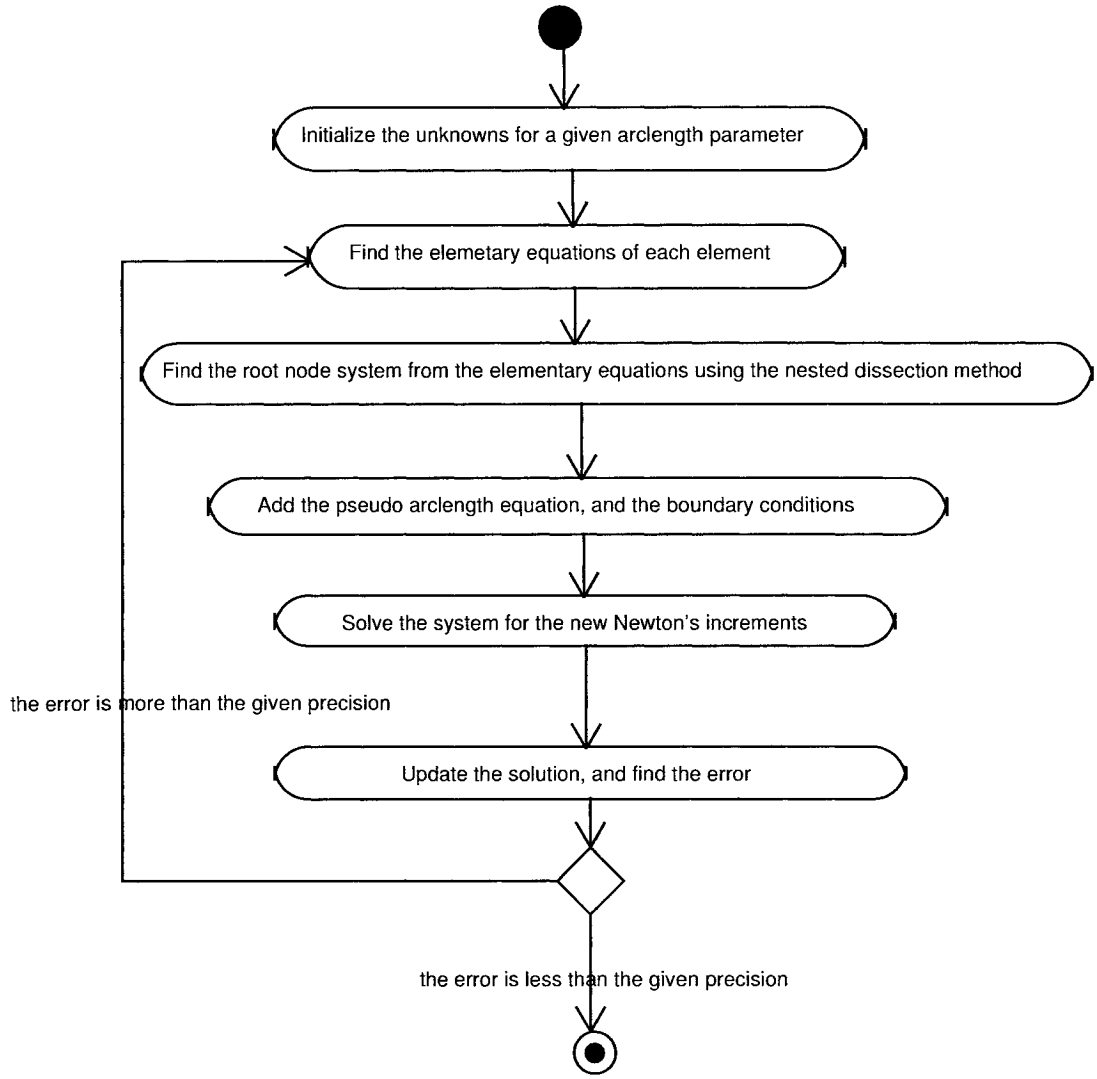


Figure 10.10: Activity diagram of the nonlinear solver object

Bifurcation Analysis object goes to the final state.

10.7 Different activities of solving a nonlinear system

In Section 9.5, an algorithm of the finite elements collocation method for systems of nonlinear elliptic PDE with a parameter-dependent integral boundary condition is given. Using this algorithm, we present different activities of the action "solving the system" of the "Continuation" state, presented in Section 10.6. These activities are demonstrated in the activity diagram of Figure 10.10. Note, an activity diagram is a specific case of a state diagram in which all (or at least most) of the states are

action states, and in which all (or at least most) of the transitions are triggered by completion of the previous actions. The aim of this diagram is to focus on flows of internal processing (contrary to external events). An action state is a state with an internal action and at least one outflow transition indicating the event of completing the internal action. In this diagram activities are the performance of operations and the transitions are motivated by the completion of the operations.

As shown in Figure 10.10, the first activity is the initialization of the unknowns, *i.e.*, initialization of $\bar{u}$, $\bar{v}$, $\hat{c}$ and λ of the mentioned algorithm, for a given value of the arclength parameter s and the step size Δs. The second activity is finding the elementary equations of each element, *i.e.*, finding A, B, d and $\hat{d}$ of Algorithm 9.5. The third activity is using the nested dissection method to constructing the system of equations of the root node of binary tree from the elementary equations. In the fourth activity the pseudo-arclength equation, Equation (9.24), and the boundary conditions, Equation (9.25), are added to the system. Finally in the fifth activity, the system must be solved for Newton's increments. In the next activity, the solutions must be updated, and the relative error must be found. If the error is less than the acceptable precision requested by the user we go to the final activity, if not we return back to the second activity.

10.8 Structure of the software

In this section, we consider some implementation aspects, including the source code structure and the run-time implementation structure. We use implementation diagrams to present these aspects. The implementation diagrams are in two forms: the component diagrams that show the structure of the code and the deployment diagrams that show the structure of the run-time system. A component diagram is a graph of components connected by dependency relationships. A deployment diagram is used to present the configuration of run-time processing elements and software components, the processes, and the objects that execute on them. A component is shown as a rectangle with two small rectangles protruding from its side.

In the component diagram of Figure 10.11 the dependencies among software components are presented. The executable component "PDEs Bifurcation Analysis" needs the following components: "Linear Algebra Library", "Other .obj files", and

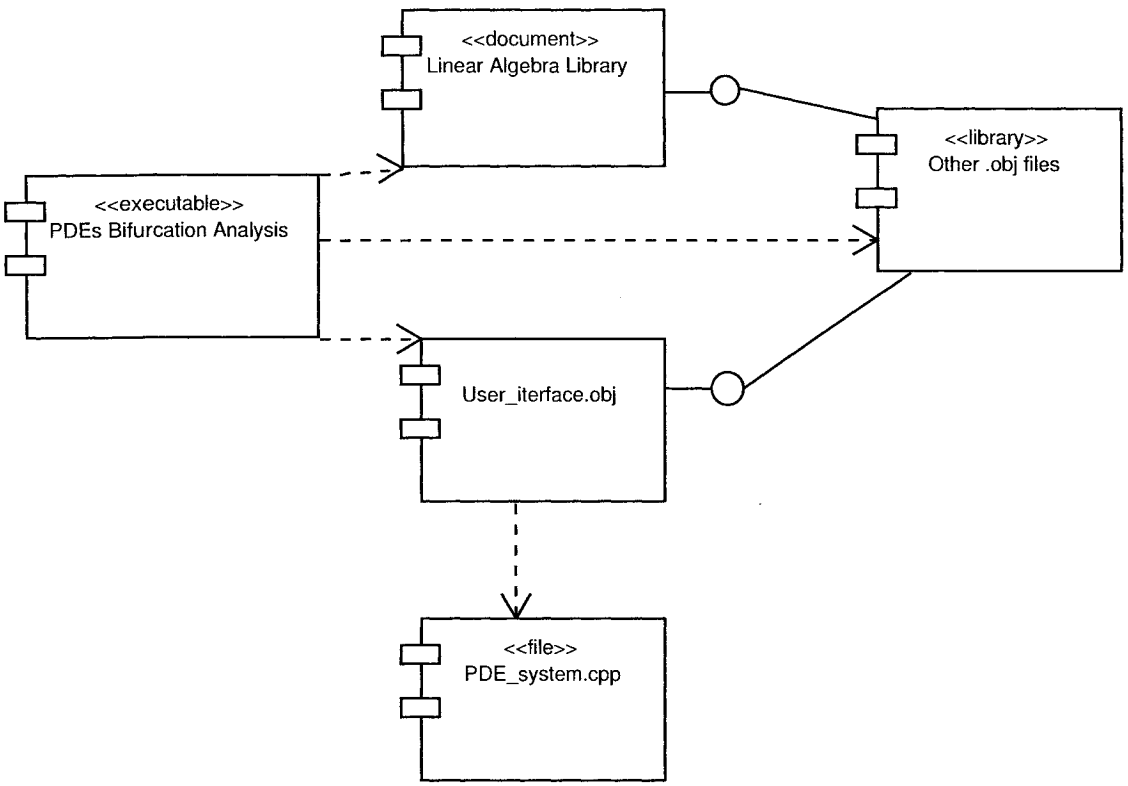


Figure 10.11: Component diagram of the software

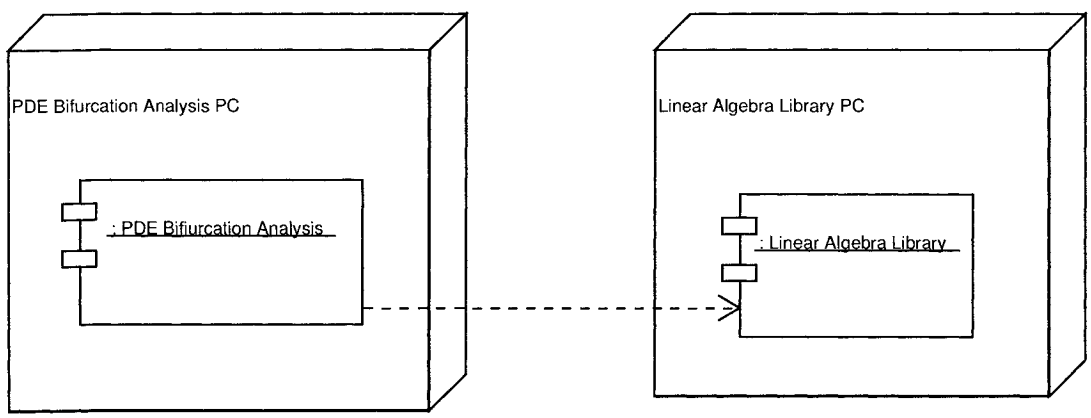


Figure 10.12: Deployment diagram of a PDE bifurcation analysis

"User_interface.obj". In our prototype software, the PDEs system is written in the file "PDE_system.cpp". This file must be added to the file "User_interface.cpp" and "User_interface.cpp" must be compiled to make "User_interface.obj". As one can see, there is a dependency between "PDE_system.cpp" and "User_interface.obj". The object file "User_interface.obj" must link to the other objects and libraries to make the executable.

The library component "Other .obj files" is a home made library, which constructed from the object files of our defined classes, *i.e.*, classes such as "Region", "Element", "Pseudo-Arclength" and so on. The links between the component "Other .obj files" and the interface of the component "Linear Algebra Library" shows the utilization of the linear algebra operations by the component "Other .obj files". Similarly, the link between the component "Other .obj files" and the interface of "User_interface.obj" shows that .obj files of the component "Other .obj files" can access to the information store in the component "User_interface.obj".

The component "Linear Algebra Library" can be linked at the run-time to the executable component. Therefore, it can be placed on another *node*, for example on the sever computer. Note here, a node is a run-time physical object that represents a processing resource. This situation is shown on the deployment diagram of Figure 10.12. As it is shown in the figure the component "PDE Bifurcation analysis" is executed on the node "PDE Bifurcation Analysis PC". This component can use the DLL library, which is placed on another computer, here, the library "Linear Algebra Library", which is on the node "Linear Algebra Library PC".

Chapter 11

Numerical Applications

11.1 A simple PDE with known solution

First, we use our collocation method for solving a PDE with a known analytical solution. Consider the following PDE

$$\Delta u = (2x^2y^2 + 2x^2y + 2xy^2 - 6xy)e^{x+y} \quad \text{for} \quad (x, y) \in \Omega \subset R^2,$$

$$u = 0 \quad \text{on} \quad \delta\Omega,$$

where Δ is the Laplace operator, the domain (Ω) is a unit square, and $(\delta\Omega)$ denotes the boundary of Ω. The analytical solution is

$$u(x, y) = x(x - 1)y(y - 1)e^{x+y}.$$

We do not consider the continuation of solutions in this example. Our objective is to compare the numerical solution, found with our method, with the analytical solution. Figure 11.1 represents the analytical solution of the above PDE. The approximate numerical solutions of this PDE, for different mesh sizes and collocation points, are shown in Figures 11.2 to 11.5. It is clear from these figures that the approximate

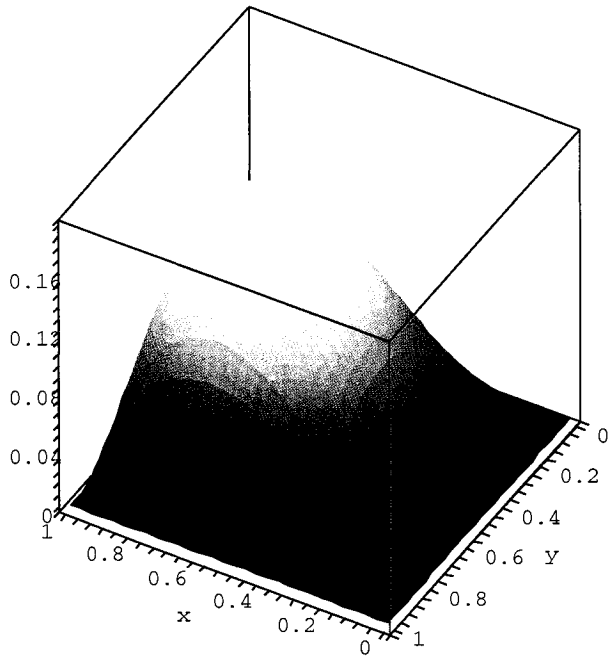
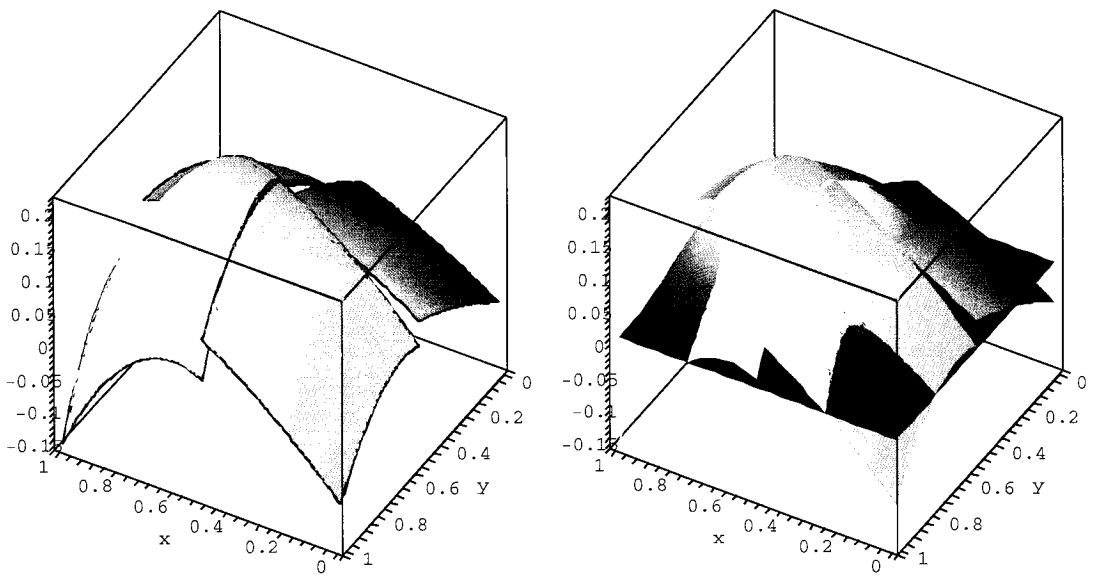


Figure 11.1: The function $u(x, y) = x(x-1)y(y-1)e^{x+y}$

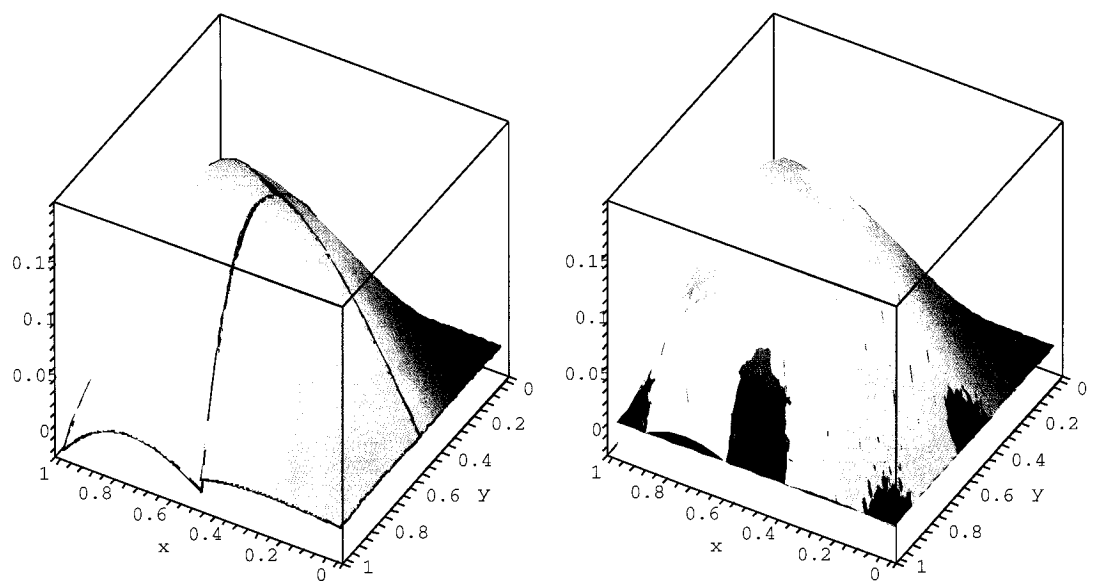
solution converges to the exact solution when the mesh size is reduced, or when the number of collocation points per element increases. As one can see, in Figure 11.2a, for a 2 by 2 mesh and only one collocation point per element, there are gaps between element boundaries. Nevertheless, the boundary lines are connected to each other at the matching points. We have the same solution value at these points, due to the matching conditions between adjacent elements. The gaps are reduced in size by increasing the number of matching points, see Figure 11.3, or by increasing the number of elements, see Figures 11.4 and 11.5.

Table 11.1 presents the maximum error at the matching points for different meshes, and different number of collocation points per element. The collocation and matching points are chosen at the Gauss points. The convergence rates between consecutive values in each column are shown in parentheses. As one can see, for $2 * 2$ $(= 4)$ collocation points per element, by reducing the mesh size we approach $\mathcal{O}(h^4)$ convergence, as predicted in Section 5.9.



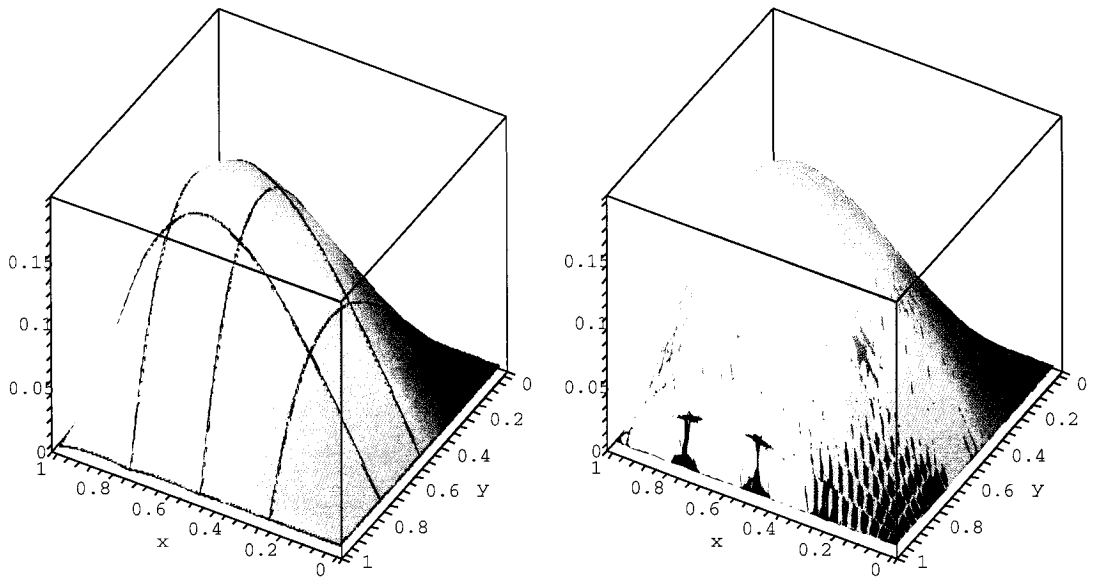
a *b*

Figure 11.2: A 2 by 2 mesh with one collocation point per element. a) The approximate solution, b) Comparing the approximate solution with the actual solution



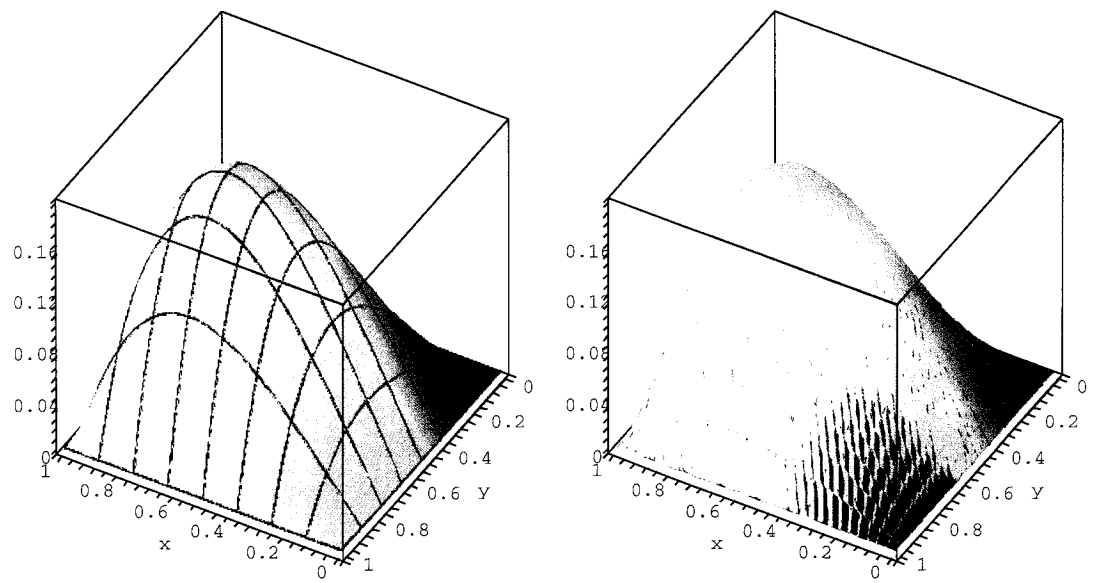
a *b*

Figure 11.3: A 2 by 2 mesh with 4 collocation points per element. a) The approximate solution, b) Comparing the approximate solution with the actual solution



a *b*

Figure 11.4: A 4 by 4 mesh with 4 collocation points per element. a) The approximate solution, b) Comparing the approximate solution with the actual solution



a *b*

Figure 11.5: An 8 by 8 mesh with 4 collocation points per element. a) The approximate solution, b) Comparing the approximate solution with the actual solution

# col. mesh	1 max. error (o)	2 * 2 max. error (o)	3 * 3 max. error (o)	4 * 4 max. error (o)
2 * 2	$8.84e^{-3}$	$4.54e^{-4}$	$2.09e^{-5}$	$4.04e^{-7}$
4 * 4	$2.80e^{-3}$ (1.66)	$5.18e^{-5}$ (3.13)	$7.64e^{-7}$ (4.78)	$7.40e^{-9}$ (5.77)
8 * 8	$7.21e^{-4}$ (1.96)	$4.30e^{-6}$ (3.59)	$2.72e^{-8}$ (4.81)	$1.32e^{-10}$ (5.81)
16 * 16	$1.81e^{-4}$ (1.99)	$3.02e^{-7}$ (3.83)	$8.84e^{-10}$ (4.95)	$2.14e^{-12}$ (5.95)
32 * 32	$4.54e^{-5}$ (2.00)	$1.99e^{-8}$ (3.92)	$2.81e^{-11}$ (4.97)	$3.40e^{-14}$ (5.98)
64 * 64	$1.14e^{-5}$ (2.00)	$1.28e^{-9}$ (3.96)	$8.87e^{-13}$ (4.98)	$5.55e^{-16}$ (5.94)

Table 11.1: The maximum error, and the order of convergence at the matching points

11.2 The Bratu-Gelfand Problem

The Bratu-Gelfand problem in $2D$ is an interesting nonlinear PDE problem, where the location of its fold, for higher-dimensional problems and more general domains, is still a subject of investigation [69, 24]. This problem can be written as

$$\Delta u + \lambda e^u = 0 \quad \text{for} \quad (x, y) \in \Omega, \tag{11.1}$$

$$u = 0 \quad \text{on} \quad \delta\Omega,$$

where the domain Ω is a unit square and λ is the continuation parameter.

The above nonlinear PDE is solved using our prototype software. Table 11.2 presents the location of the fold for various choices of the mesh sizes and the collocation points. Using this table, the exact location of the fold is assumed to be the best value found in the table, *i.e.*, $\lambda = 6.808124423$. Table 11.3 shows the maximum error of the fold position, with respect to the assumed position, for different mesh sizes and collocation points. Between parentheses, the convergence rate with respect to the previous value is given. As in the previous example in Section 11.1, we have $\mathcal{O}(h^2)$ convergence for 1 collocation point per element, and $\mathcal{O}(h^4)$ convergence for 4 collocation points per element.

Figure 11.6 presents the solution surfaces at the fold position for different meshes and collocation points. Figure 11.7 shows the bifurcation diagram of the Bratu-Gelfand problem. There is only one fold in the $2D$ Bratu-Gelfand problem. Continuing the solution family, the norm of the solutions goes to infinity as λ approaches to

# *col.*	1	$2*2$	$3*3$	$4*4$
mesh	λ	λ	λ	λ
$1*1$	5.886071059	8.829106588	6.468997582	6.896291085
$2*2$	7.848094745	6.657615709	6.819161991	6.807812085
$4*4$	6.861418190	6.805822324	6.808174115	6.808122960
$8*8$	6.820207335	6.808016922	6.808124690	6.808124424
$16*16$	6.811079454	6.808117899	6.808124427	6.808124423
$32*32$	6.808859009	6.808124018	6.808124423	6.808124423
$64*64$	6.808307808	6.808124397	6.808124423	6.808124423

Table 11.2: The location of the fold in the Bratu-Gelfand problem

# *nb.col.*	1	$2*2$	$3*3$	$4*4$
mesh	*max. error* (o)	*max. error* (o)	*max. error* (o)	*max. error* (o)
$1*1$	$9.22e^{-1}$	2.02	$3.39e^{-1}$	$8.82e^{-2}$
$2*2$	1.04 (0.17)	$1.51e^{-1}$ (3.75)	$1.10e^{-2}$ (4.94)	$3.12e^{-4}$ (8.14)
$4*4$	$5.33e^{-2}$ (4.29)	$2.30e^{-3}$ (6.03)	$4.97e^{-5}$ (7.80)	$1.46e^{-6}$ (7.74)
$8*8$	$1.21e^{-2}$ (2.14)	$1.08e^{-4}$ (4.42)	$2.67e^{-7}$ (7.54)	$1.00e^{-9}$ (10.5)
$16*16$	$2.96e^{-3}$ (2.03)	$6.52e^{-6}$ (4.04)	$4.00e^{-9}$ (6.06)	
$32*32$	$7.35e^{-4}$ (2.01)	$4.05e^{-7}$ (4.01)		
$64*64$	$1.83e^{-4}$ (2.00)	$2.60e^{-8}$ (3.96)		

Table 11.3: Maximum error at the matching points in the Bratu-Gelfand problem

zero. Figure 11.8 shows solutions of the Bratu-Gelfand problem for different values of continuation parameter λ. The solutions before the fold, at the fold, and the after the fold point, drawn on the same scale, are presented in this figure.

11.3 Bifurcation example 1

In this section, we consider a simple linear PDE, having known bifurcation branches, namely those that correspond to eigenvalues. This is useful to check the branch point detection functionality, and the branch switching capability of the prototype software. Consider the following PDE

$$\Delta u + \lambda u = 0 \quad \text{for} \quad (x, y) \in \Omega, \tag{11.2}$$

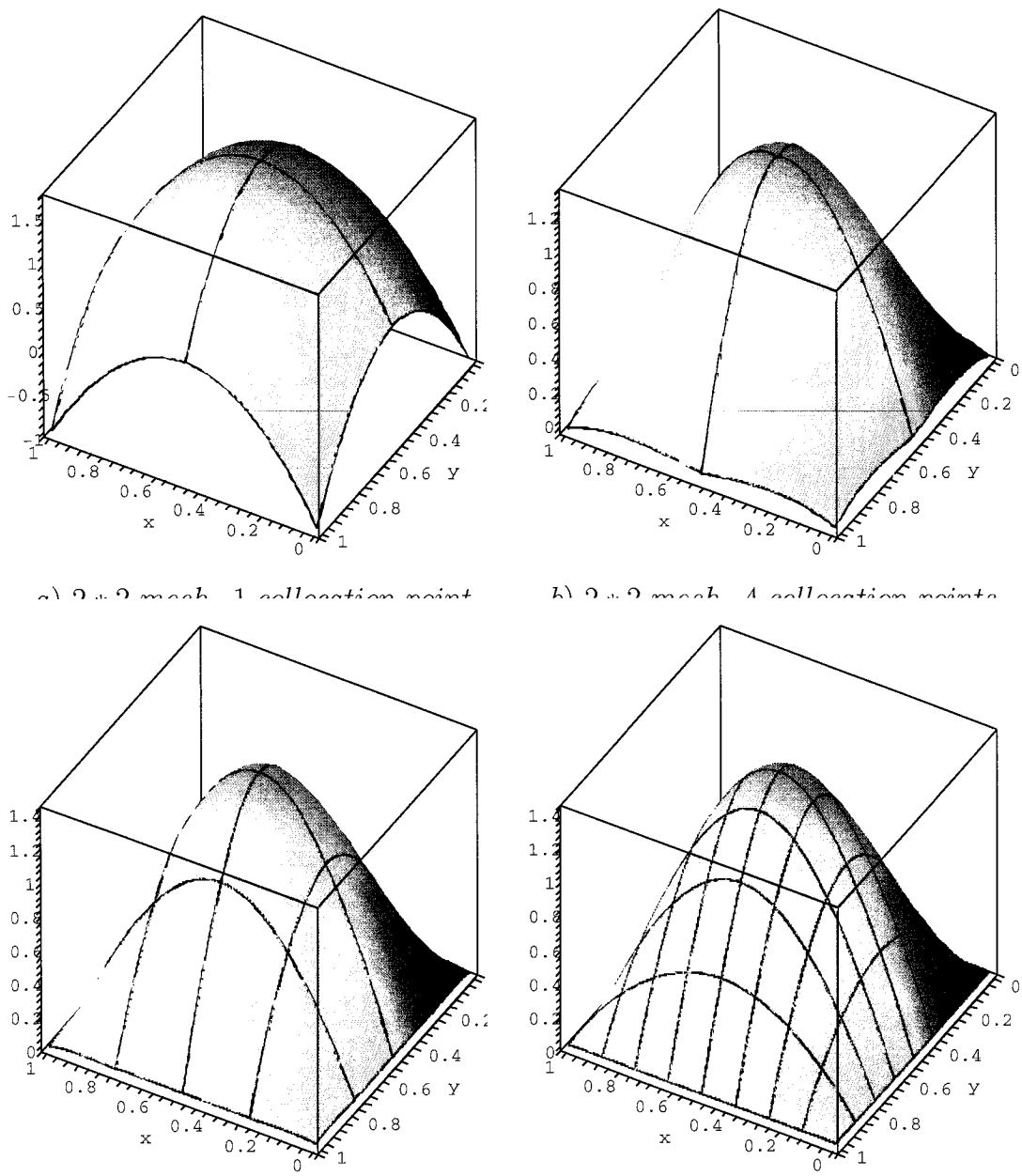
$$u = 0 \quad \text{on} \quad \delta\Omega,$$

where the domain Ω is a unit square and λ is the continuation parameter.

This equation corresponds to an eigenvalue problem, which has nontrivial solutions at $\lambda = 2k^2\pi^2$, where k is a positive integer. The nontrivial solutions are

$$u(x, y) = \sin(\pi k x) \sin(\pi k y).$$

The first branch point is located at $2\pi^2 = 19.73920880$. Figure 11.9 shows the bifurcation diagram with the first bifurcating branch, as found by our prototype software, using a 4 by 4 mesh and 4 collocation points per element. Table 11.4 illustrates the location the first branch point (eigenvalue) in this problem for various choices of meshes and collocation points. Table 11.5 presents the maximum error in the detection of the first branch point location. As before, the convergence rates between two consecutive values are given in parenthesis. Figure 11.10 shows several solutions (eigenfunctions) on the vertical bifurcation branch.



a) $2 * 2$ *mesh, 1 collocation point*

b) $2 * 2$ *mesh, 4 collocation points*

c) $4 * 4$ *mesh, 4 collocation points*

d) $8 * 8$ *mesh, 4 collocation points*

Figure 11.6: Solution of the Bratu-Gelfand problem at the fold, for different mesh sizes and collocation points per element

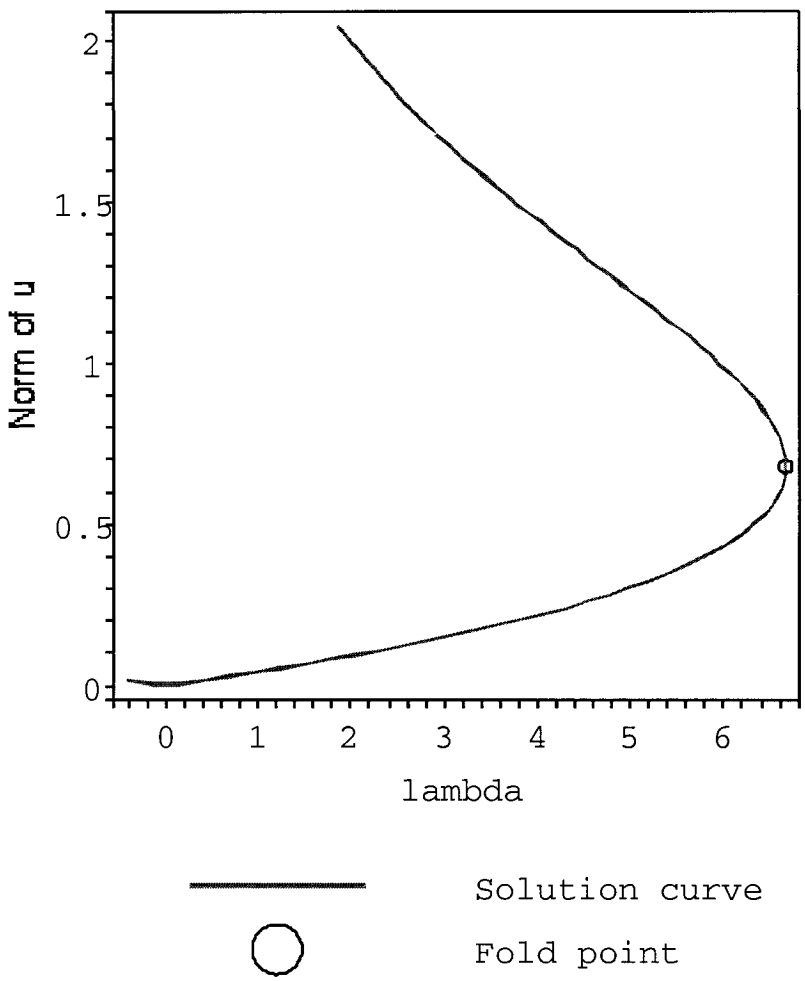
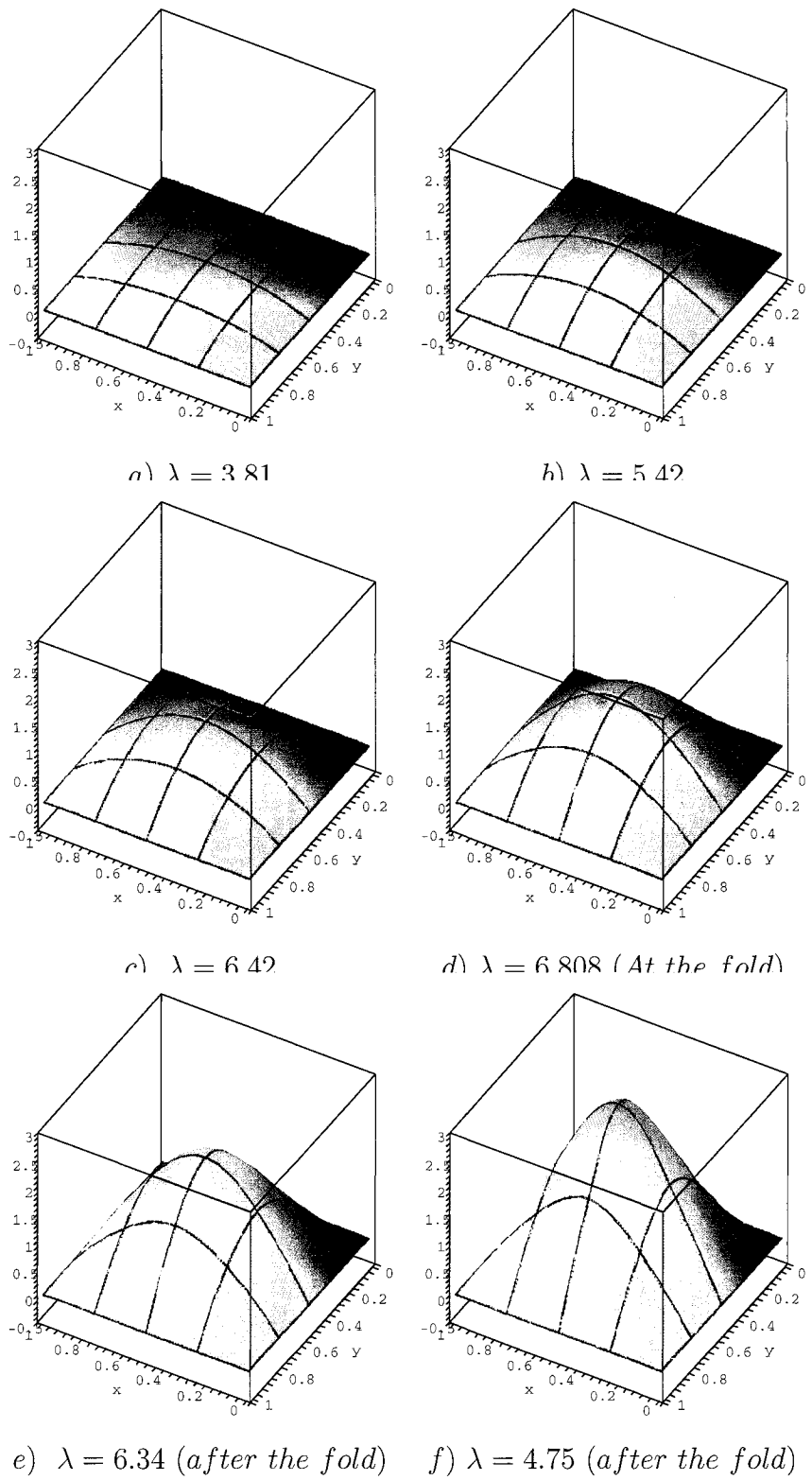


Figure 11.7: Bifurcation diagram of the Bratu-Gelfand problem

# col.	1	$2*2$	$3*3$	$4*4$
mesh	λ	λ	λ	λ
$2*2$	21.33333333	19.78370023	19.73985662	19.73921407
$4*4$	20.22619483	19.74251591	19.73921994	19.73920882
$8*8$	19.86487048	19.73942347	19.73920898	19.73920880
$16*16$	19.77084568	19.73922234	19.73920880	19.73920880
$32*32$	19.74713151	19.73920965	19.73920880	19.73920880

Table 11.4: The location of the first branch point of $\Delta u + \lambda u = 0$



a) $\lambda = 3.81$ b) $\lambda = 5.42$

c) $\lambda = 6.42$ d) $\lambda = 6.808$ *(At the fold)*

e) $\lambda = 6.34$ *(after the fold)* f) $\lambda = 4.75$ *(after the fold)*

Figure 11.8: Solution of the Bratu-Gelfand problem for different values of λ, using a 4 by 4 mesh and 4 collocation points per element

# *col.*	1	$2*2$	$3*3$	$4*4$
mesh	*max. error* (o)	*max. error* (o)	*max. error* (o)	*max. error* (o)
$2*2$	1.59	$4.45e^{-2}$	$6.48e^{-4}$	$5.27e^{-6}$
$4*4$	$4.87e^{-1}$ (1.71)	$3.31e^{-3}$ (3.75)	$1.11e^{-5}$ (5.87)	$2.00e^{-8}$ (8.04)
$8*8$	$1.26e^{-1}$ (1.95)	$2.15e^{-4}$ (3.94)	$1.80e^{-7}$ (5.95)	0
$16*16$	$3.16e^{-2}$ (2.00)	$1.35e^{-5}$ (3.99)	0	0
$32*32$	$7.92e^{-3}$ (2.00)	$8.50e^{-7}$ (3.99)	0	0

Table 11.5: Error and order of convergence in the calculation of the first branch point of $\Delta u + \lambda u = 0$

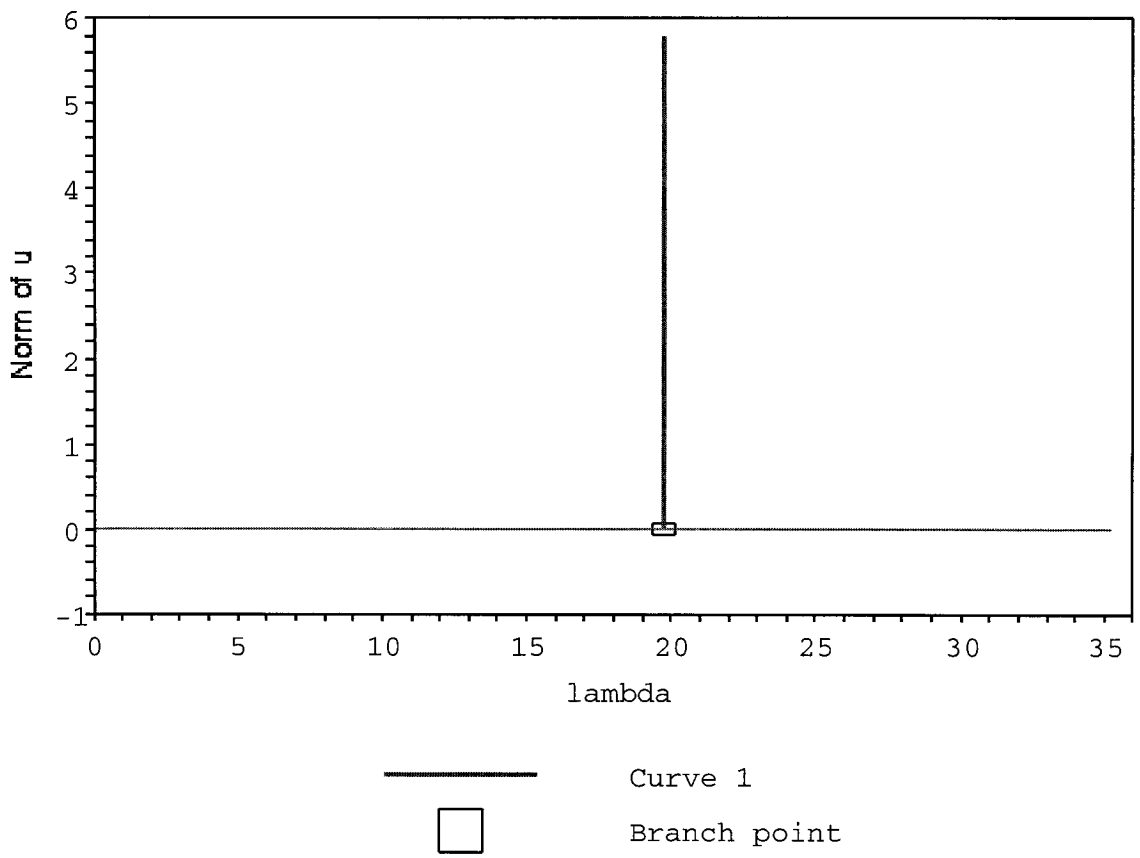
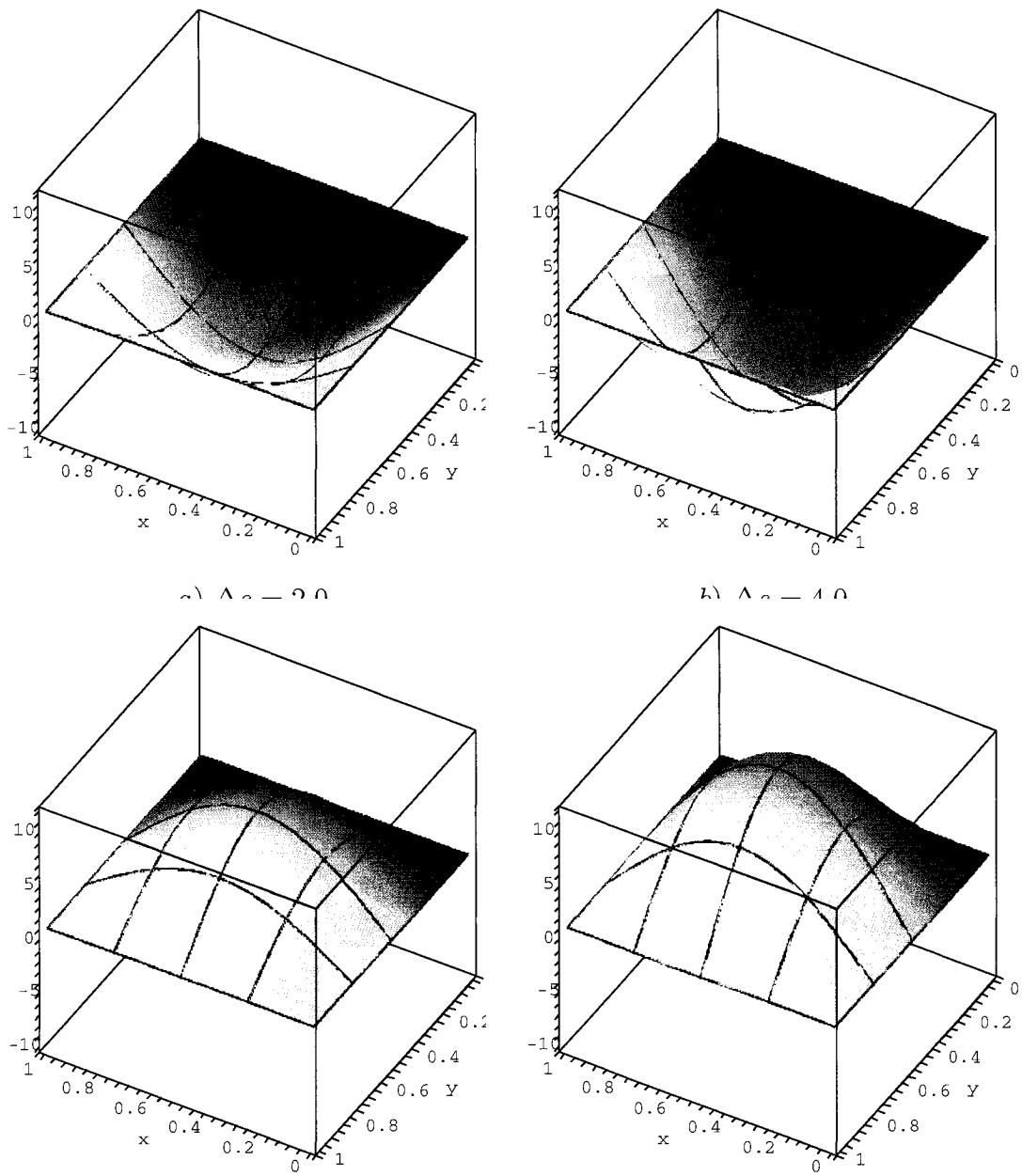


Figure 11.9: Bifurcation diagram of $\Delta u + \lambda u = 0$



c) $\Delta s = -2.0$ d) $\Delta s = -4.0$

Figure 11.10: Solutions along the vertical branch of $\Delta u + \lambda u = 0$ at $\lambda = 19.73920880$. The mesh is 4 by 4, with 4 collocation points per element. Δs is the distance from the trivial solution

11.4 Bifurcation example 2

In this section, we consider a bifurcation problem, which has curved bifurcating branches. Consider the following PDE

$$\Delta u + \lambda \sin(u) = 0 \quad \text{for} \quad (x, y) \in \Omega, \tag{11.3}$$

$$u = 0 \quad \text{on} \quad \delta\Omega,$$

where the domain Ω is a unit square and λ is the continuation parameter.

Figure 11.11 shows the bifurcation diagram with the first bifurcating branch, as computed by our software, using a 4 by 4 mesh and 4 collocation points per element. As one can see, the first branch point is located at $\lambda = 19.73920880$. This is the same value that we found in Section 11.3, because Equation (11.2) is the linearization of Equation (11.3) about $u = 0$. Therefore the location of the branch points along the trivial solution of the equation $\Delta u + \lambda \sin(u) = 0$ are the same as for the equation $\Delta u + \lambda u = 0$. Figure 11.12 shows a solution surfaces on the upper and on the lower solution branches, namely, at $\lambda = 43.0849$ and $\lambda = 55.0672$ respectively.

11.5 Bifurcation example 3

In this example, we consider a more complicated PDE, namely, a PDE with bifurcating solution branches, as well as folds on the solution branches. Consider the following PDE

$$\Delta u + \lambda(u(1 - \sin(u) + u^3) = 0 \quad \text{for} \quad (x, y) \in \Omega, \tag{11.4}$$

$$u = 0 \quad \text{on} \quad \delta\Omega,$$

where the domain Ω is the unit square and λ is the continuation parameter. This example is also investigated in a paper by Chien, Jeng and Li [24].

This problem is easily solved by our prototype software. Figure 11.13 shows the bifurcation diagram of the first bifurcating branch of this problem, using a 8 by 8 mesh and 1 collocation point per element. As one can see, the first branch point is located at $\lambda = 19.73920880$. This is the same value that we found in Sections 11.3 and 11.4, as Equation (11.2) is also the linearization of Equation (11.4) about $u = 0$,

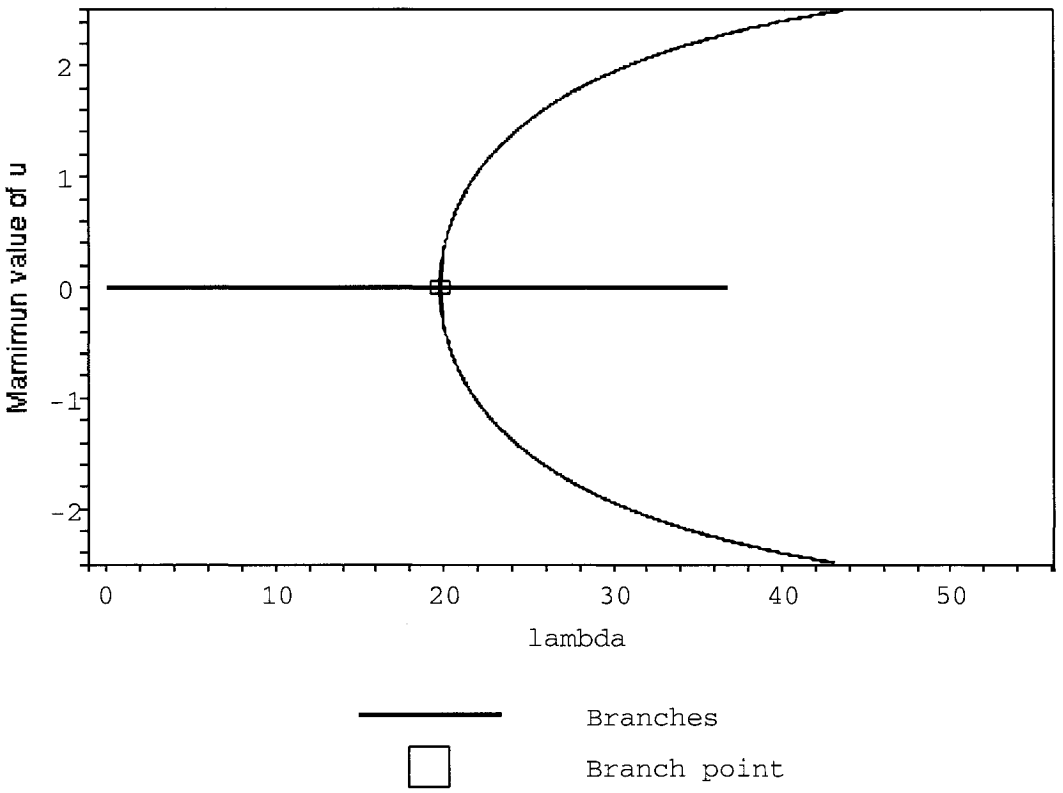
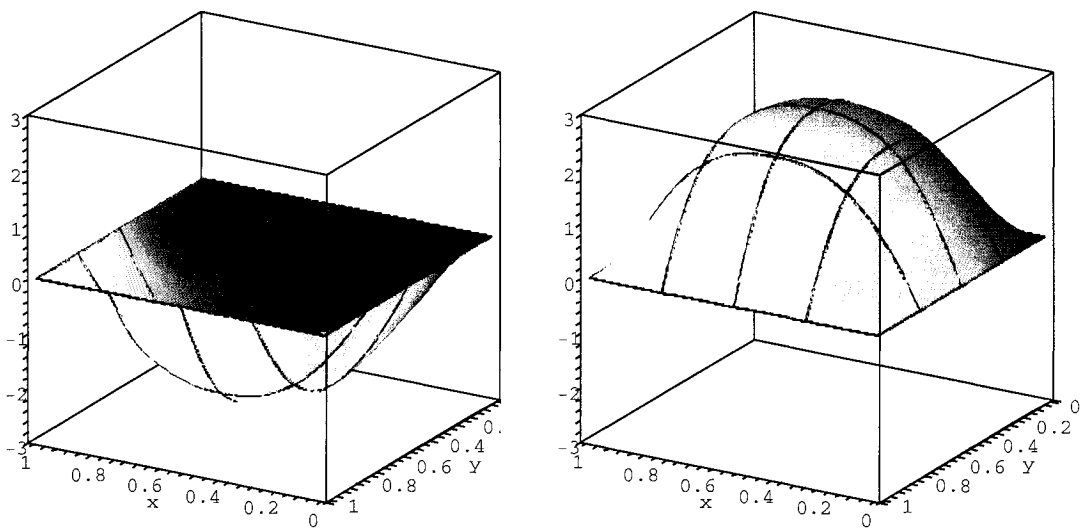


Figure 11.11: Bifurcation diagram of $\Delta u + \lambda \sin(u) = 0$



a) $\lambda = 43.0849$ *on the lower branch*

b) $\lambda = 55.0672$ *on the upper branch*

Figure 11.12: Two solutions of $\Delta u + \lambda \sin(u) = 0$: a) at $\lambda = 43.0849$ on the lower branch; and b) at $\lambda = 55.0672$ on the upper branch. The mesh is 4 by 4, with 4 collocation points per element

and $u - u^2 + u^3 \approx u$ when $u \to 0$. As a result, the position of the bifurcation points from the trivial solution are the same as in Examples 11.3 and 11.4.

Our software detects a fold on the upper solution branch of the above equation. This fold is located at $\lambda \approx 25.15$, which is the same result as in previous work on this problem. Figure 11.14 presents the solution surface at the fold for different number of collocation points per element, using the same elements. As one can see, there are gaps between the solution patches for one collocation point per element, but the solution patches are connected at the matching points. Using this figure, we can not see the gaps in case of four collocation points per elements, although these gaps still exist. Figure 11.15 shows a solution surface on the lower, and on the upper solution branches, namely, at $\lambda = 5.99$ and $\lambda = 5.877$, respectively.

11.6 The 2D Brusselator system

Reaction-diffusion systems and, in particular, the Brusselator system, arise in the study of chemical and biological phenomena. The dynamics of the Brusselator systems has been the subject of much research activity over the past decades [136, 147]. The Brusselator system is an example of an autocatalytic, oscillating chemical reaction. An autocatalytic reaction is one in which a species acts to increase the rate of its producing reaction. In many autocatalytic systems complex dynamics are seen, including multiple steady-states and periodic orbits.

In this section, we consider the $2D$ Brusselator system. This is a good example to test our prototype software for a system having 2 solution components. Consider the following PDE system

$$\begin{aligned} d_1/l^2\Delta u + (\lambda+1)u + u^2v + a &= 0 \quad \text{for} \quad (x,y) \in \Omega, \\ d_2/l^2\Delta v + \lambda u - u^2v &= 0 \end{aligned} \tag{11.5}$$

$$u = a, v = \lambda/a \quad \text{on} \quad \delta\Omega,$$

where the domain Ω is a unit square, λ is the continuation parameter, u and v represent the first and the second components of the solution. We consider the case, where

$$d_1 = 1,$$

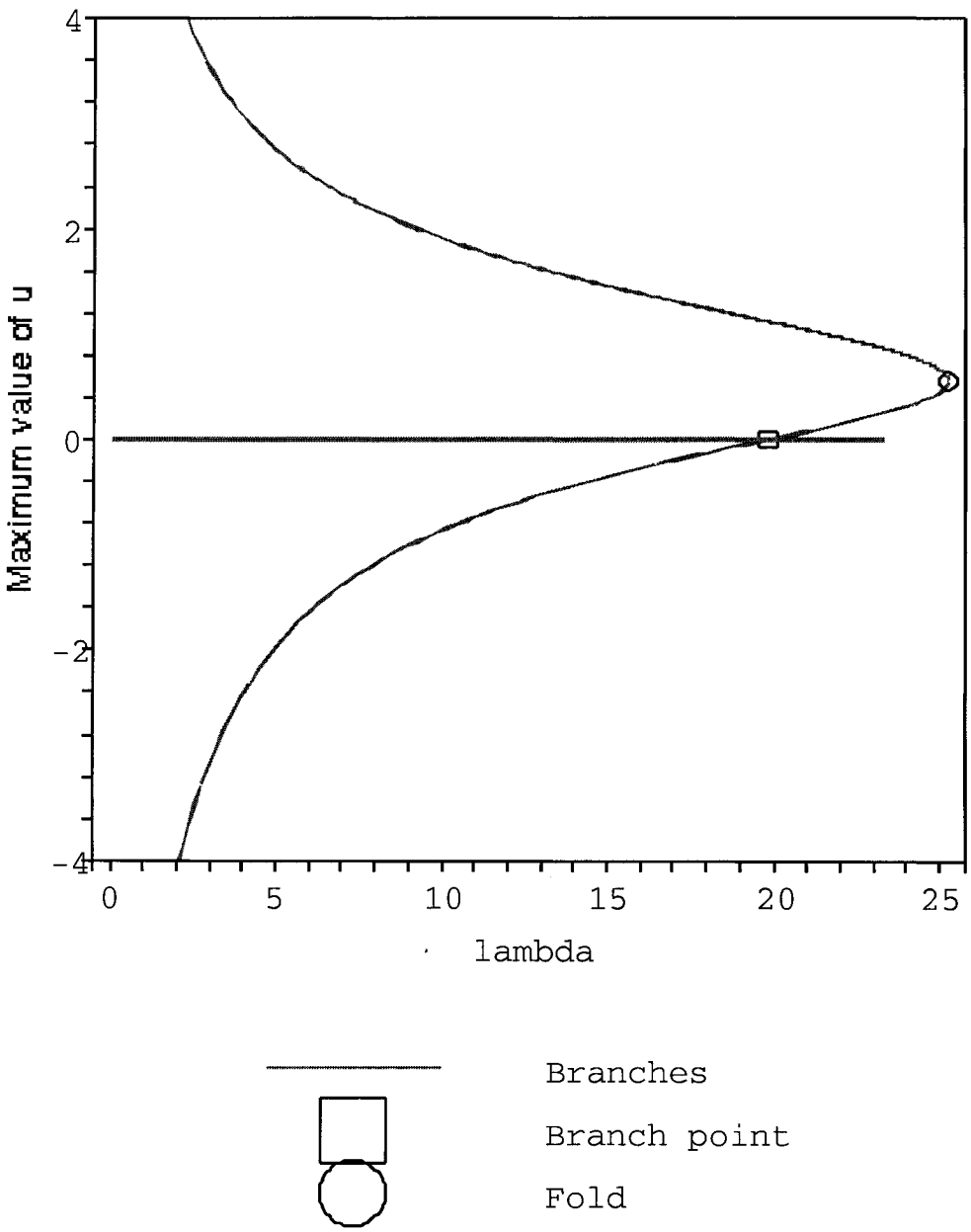
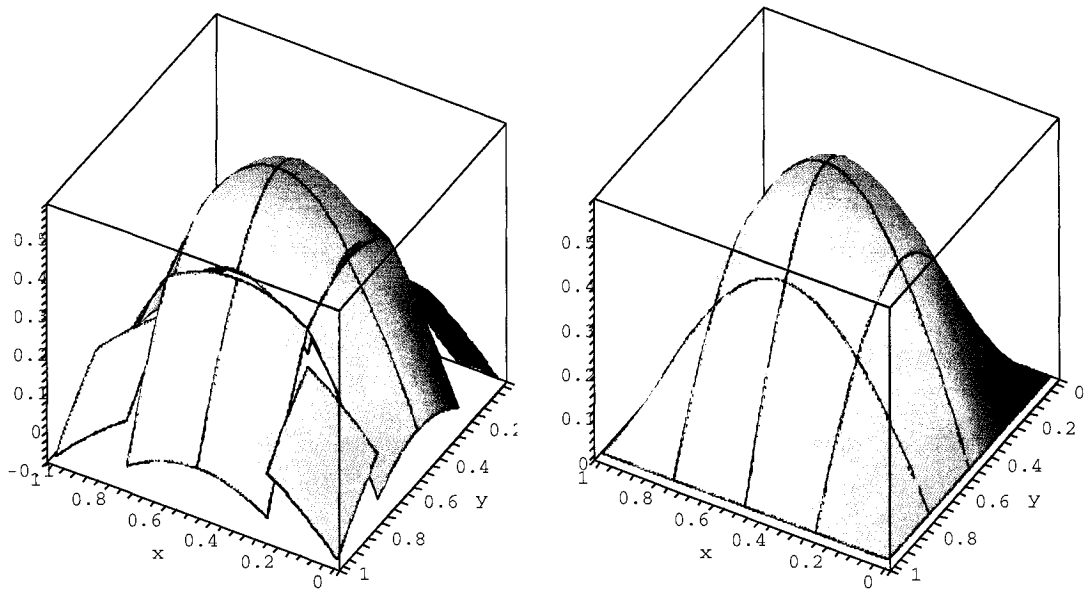
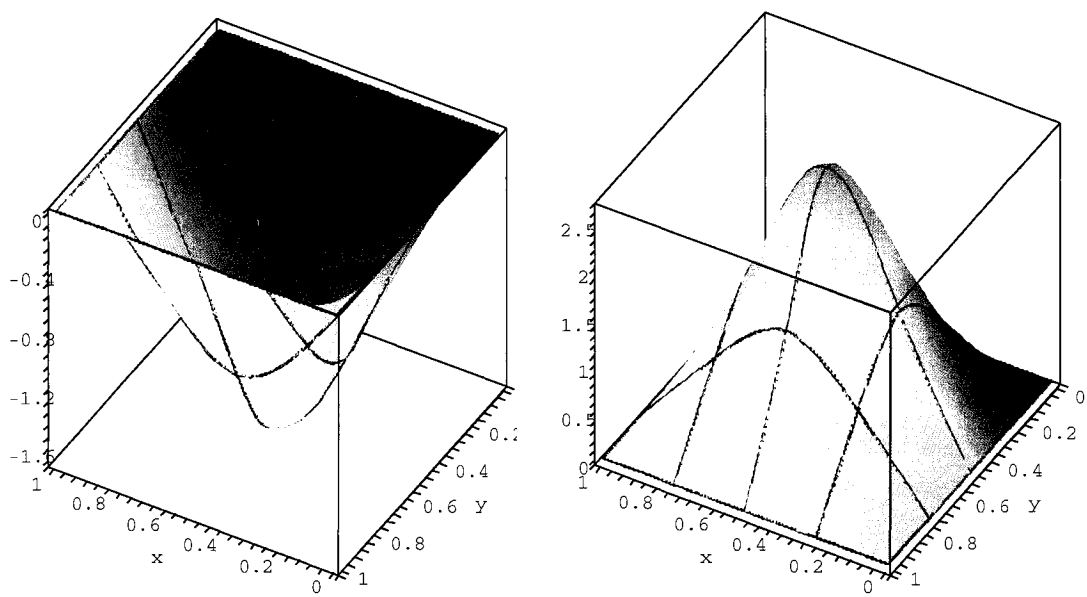


Figure 11.13: Bifurcation diagram of $\Delta u + \lambda(u(1 - \sin(u) + u^3) = 0$



a) 1 collocation point *b) 4 collocation points*

Figure 11.14: Solution at the fold of $\Delta u + \lambda(u(1 - \sin(u) + u^3) = 0$. Computed with mesh size a) 4 by 4, with 1 collocation point; and b) 4 by 4, with 4 collocation points per element



a) $\lambda = 5.99$ *on lower branch* *b)* $\lambda = 5.877$ *on upper branch*

Figure 11.15: Two solutions of $\Delta u + \lambda(u(1 - \sin(u) + u^3) = 0$ at: a) $\lambda = 5.99$, on the lower branch; and b) $\lambda = 5.877$, on the upper branch. Mesh size is 4 by 4, with 4 collocation points per element

$$
\begin{aligned}
d_2 &= 2, \\
l &= 1, \\
a &= 4.
\end{aligned} \tag{11.6}
$$

This example is also considered in a paper by Fedoseyev, Friedman and Kansa [69].

Solving this problem, using a 4 by 4 mesh, and 4 collocation point per element, we detect the first bifurcation point at $\lambda \approx 29.144$, which is close to the exact value, *i.e.*, $\lambda \approx 29.144494$. Of course, with increasing the number of elements, and increasing the number of collocation points, we can approximate the exact solution with desired precision. Figure 11.16 presents the bifurcation diagram with the first bifurcation point. In this figure the two components of solution are shown in the same diagram. Figures 11.17 and 11.18 present the bifurcation diagram with the individual solution components. As one can see, there is a fold on the upper solution branch, namely, at $\lambda \approx 26.59$.

Figures 11.19, 11.20 and 11.21 present the basic solution, the upper solution branch, and the lower solution branch, respectively. Figure 11.22 presents several solution surfaces on the basic solution branch, using a 2 by 2 mesh, and 4 collocation points per element. The pictures *c* and *d* of this figure show the branch point solutions. Figures 11.23 and 11.24 show solutions on the upper and on the lower branches, respectively. The fold point solution is presented in the pictures *c* and *d* of Figure 11.23. As we continue the bifurcating branches, the shape of the solutions changes. To be able to resolve the solution variations, we have used a 4 by 4 mesh, and 4 collocation points per element. Figures 11.25 and 11.26 present some of these solutions, for the upper and the lower branches, respectively. To continue further on the upper solution branch, and to satisfy the boundary condition at the corners of the solutions, as presented in pictures *e* and *f* of Figure 11.25, we need a finer mesh.

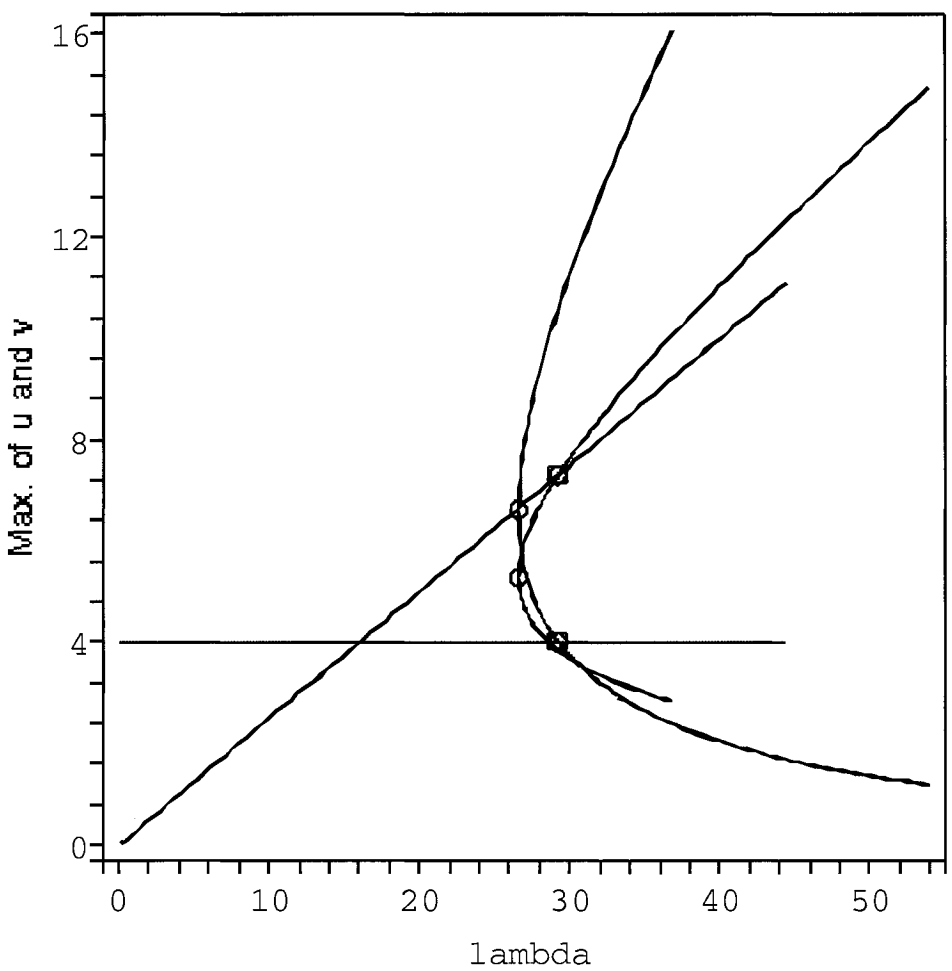


Figure 11.16: Bifurcation diagram of the Brusselator system

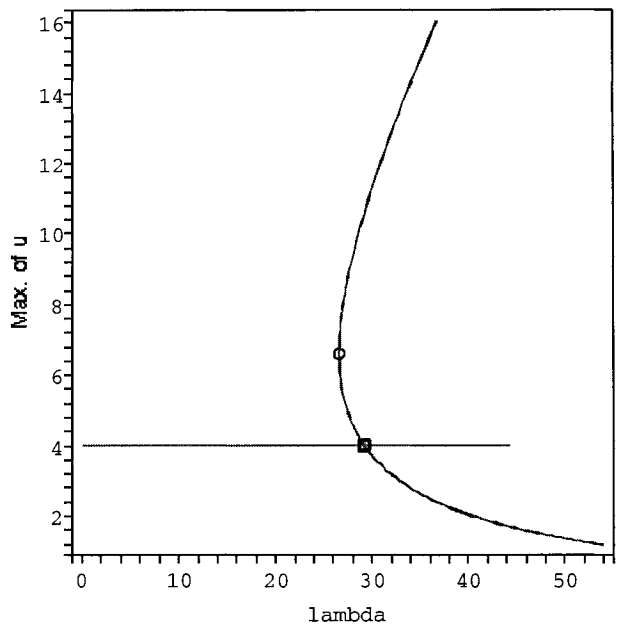


Figure 11.17: Bifurcation diagram showing the first component of the Brusselator system

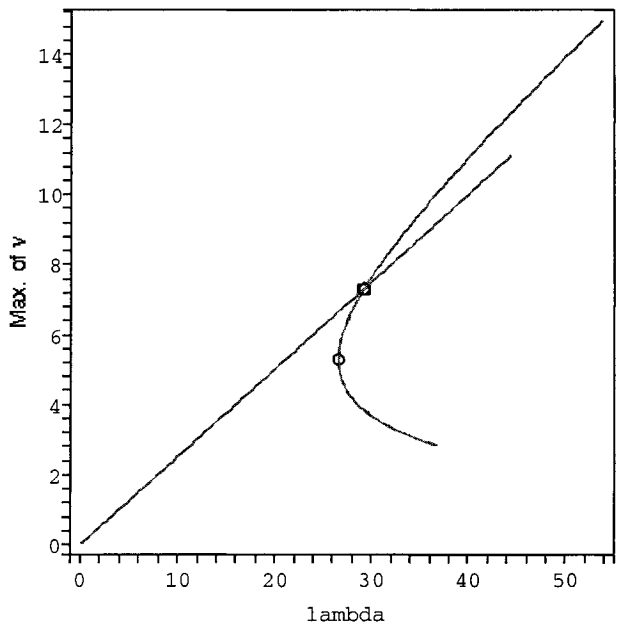


Figure 11.18: Bifurcation diagram showing the second component of the Brusselator system

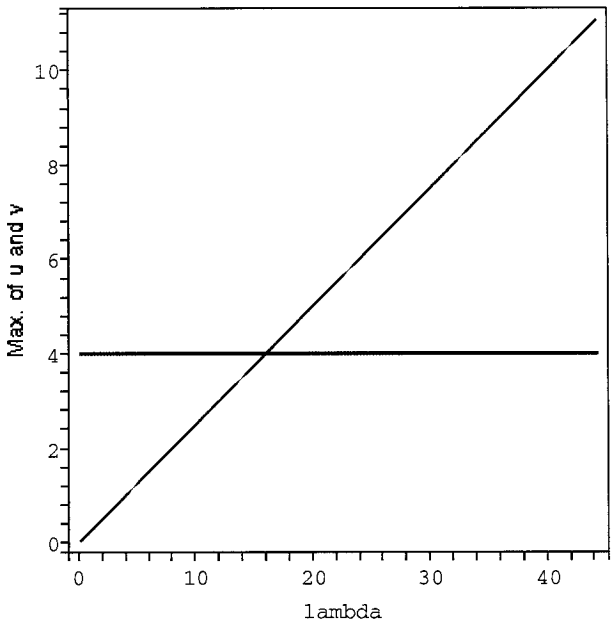


Figure 11.19: The basic solution of the Brusselator system

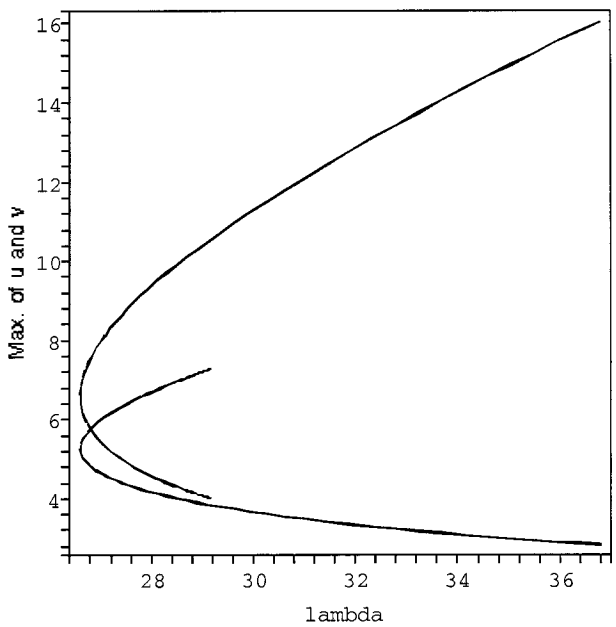


Figure 11.20: The upper solution branch of the Brusselator system. There is a fold at $\lambda = 26.59$ on this branch

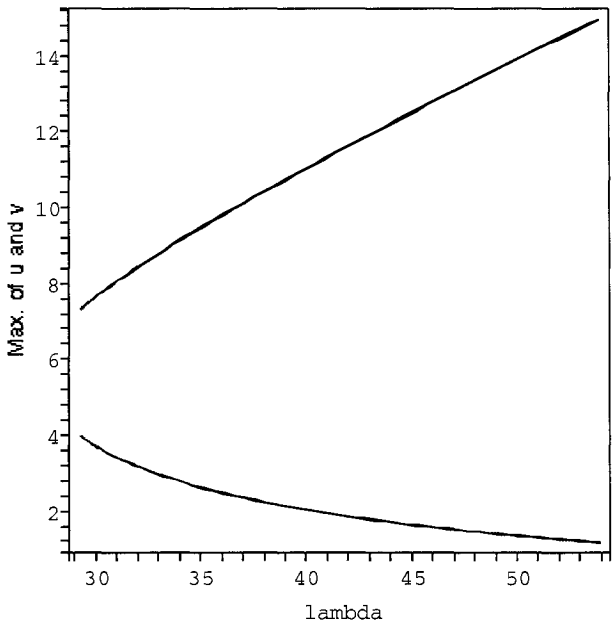
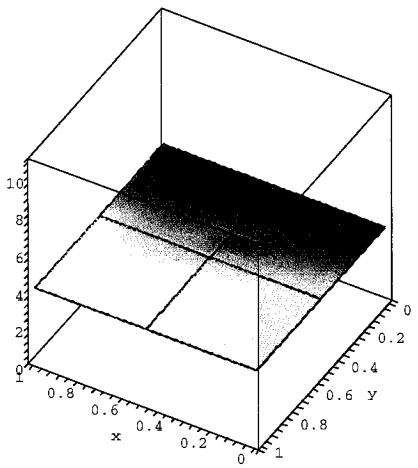


Figure 11.21: The lower solution branch of the Brusselator system

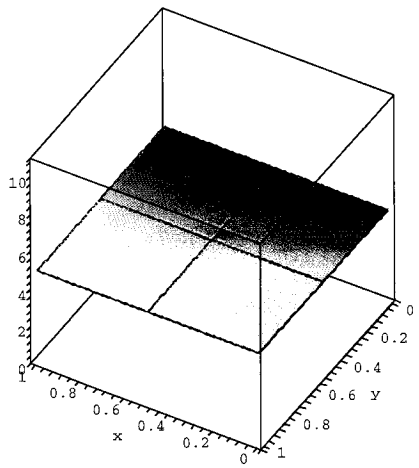
11.7 The Navier-Stokes equations

In this section we demonstrate the utility of the finite element collocation with discontinuous piecewise polynomials method, and the capability of our prototype PDEs continuation software, for solving difficult PDEs problems. As an application, we here consider the famous Navier-Stokes equations.

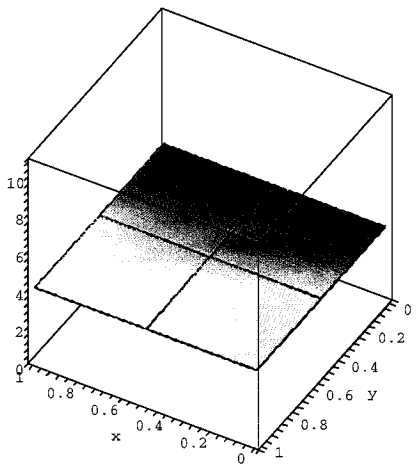
Generally, finite difference methods are used for solving simple cases of the Navier-Stokes equations. The pure finite difference method is difficult to apply, and the solution of the final system resulting from such discretization takes much execution time. In the case of using simple (low order) finite differences, a fine mesh is needed, that results in a large system of equations. In the case of higher order finite differences we encounter the problem of how to apply the boundary conditions. Hybrid methods are also considered by some authors. For example, Meyer Spasche [102, 103] has used a combination of finite differences and finite Fourier decomposition to solve the axisymmetric steady state Navier-Stokes equations that arise in the Taylor problem. Conley [33] used a combination of finite differences and Chebychev polynomials for computing the flow between two infinite parallel rotating plates.



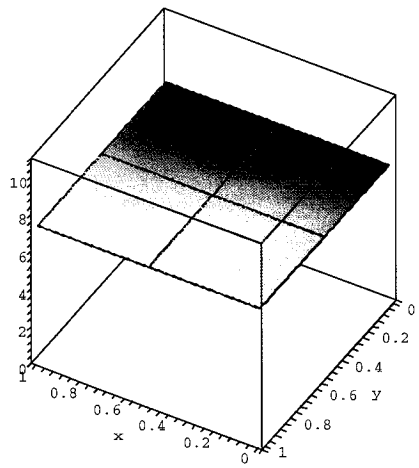
a) u for λ = 19.17



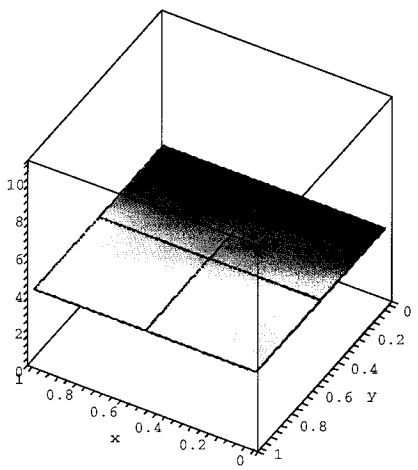
b) v for λ = 19.17



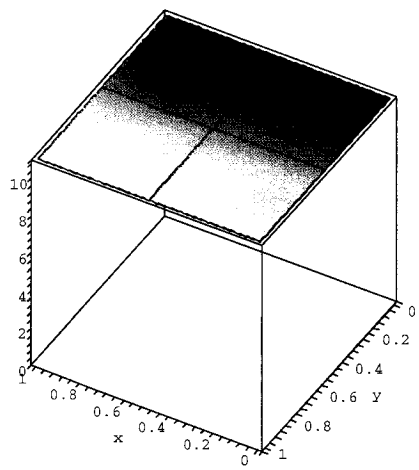
c) u for λ = 29.144 (Branch point)



d) v for λ = 29.144 (Branch point)

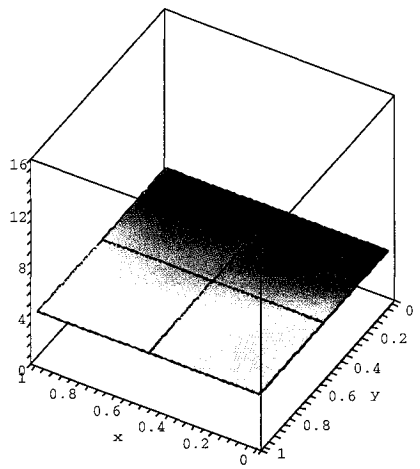


e) u for λ = 43.86

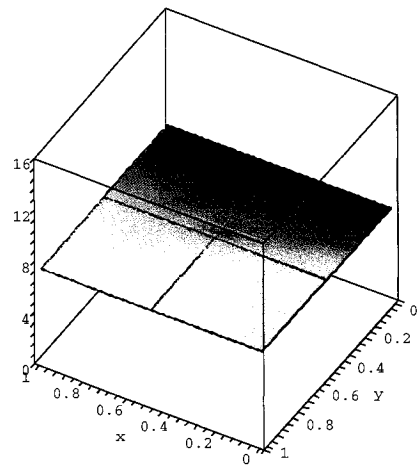


f) v for λ = 43.86

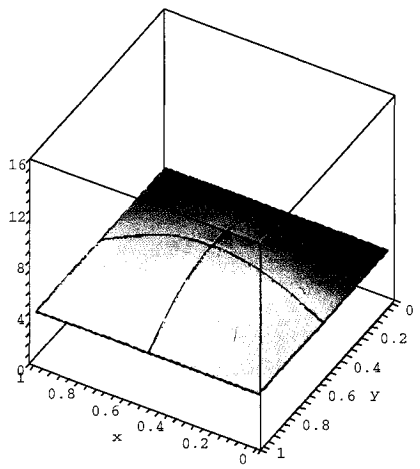
Figure 11.22: Solution of the Brusselator problem, for different values of λ on the basic solution branch, using a 2 by 2 mesh, and 4 collocation points per element. c) and d) present the solution at the branch point



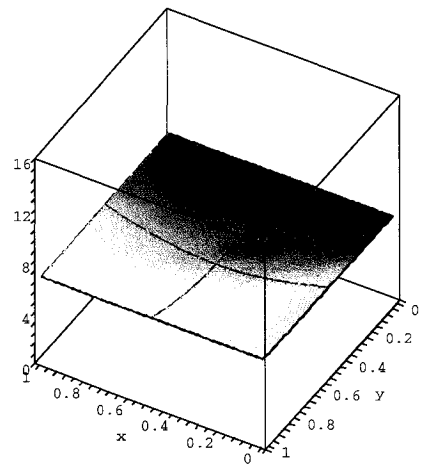
a) u for $\lambda = 29.153$



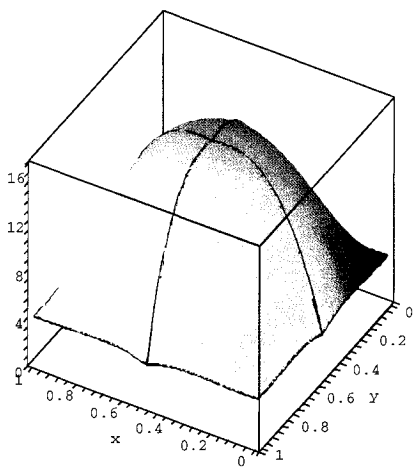
b) v for $\lambda = 29.153$



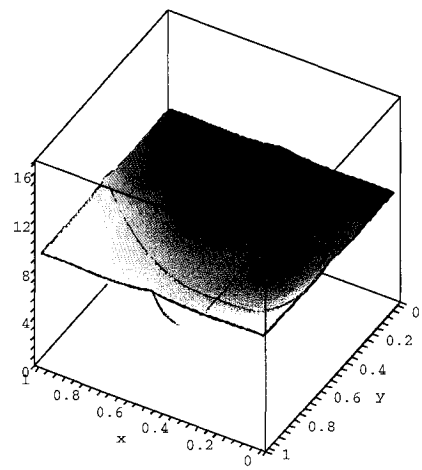
c) u for $\lambda = 26.59$ (Fold point)



d) v for $\lambda = 26.59$ (Fold point)

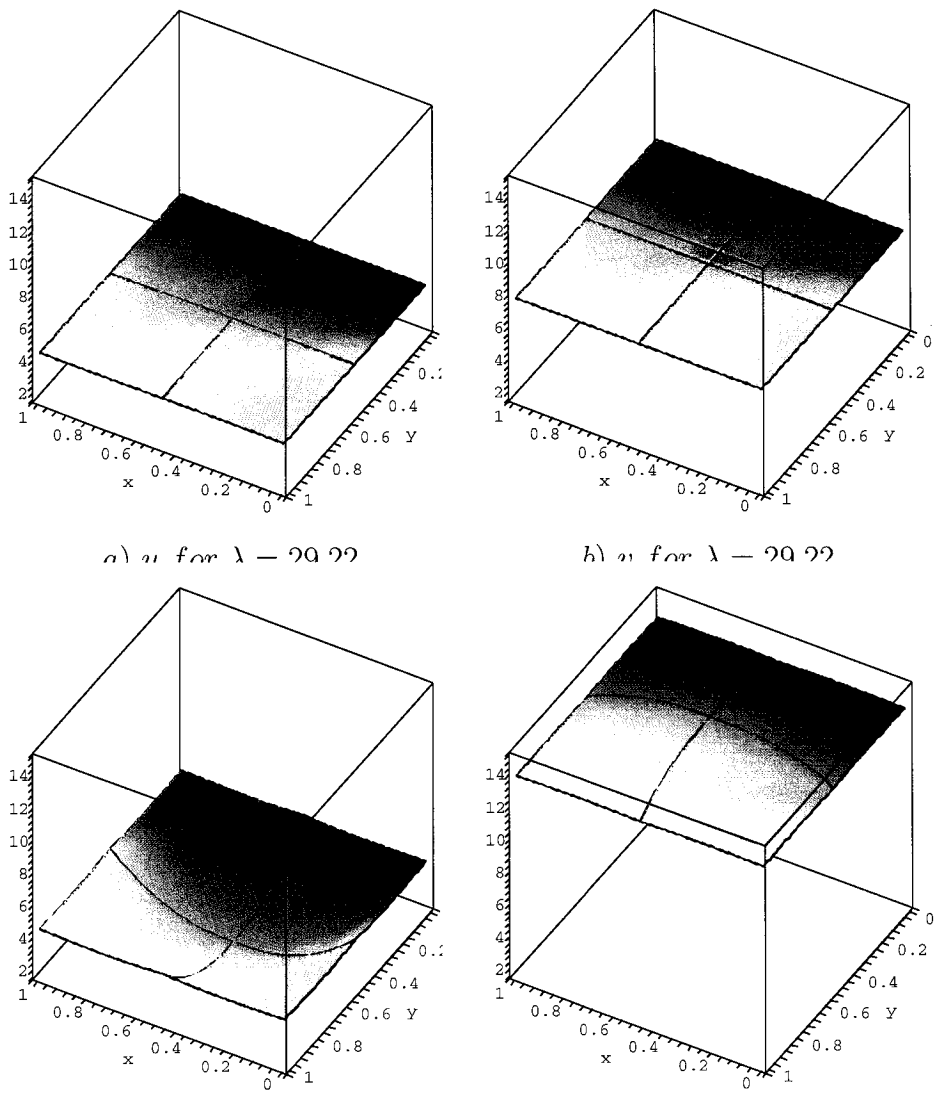


e) u for $\lambda = 36.80$



f) v for $\lambda = 36.80$

Figure 11.23: Solution of the Brusselator problem, for different values of λ on the upper solution branch, using a 2 by 2 mesh, and 4 collocation points per element. c) and d) present the solution at the fold point



a) u for λ = 29.22 *b) v for λ = 29.22*

c) u for λ = 53.92 *d) v for λ = 53.92*

Figure 11.24: Solution of the Brusselator problem for different values of λ on the lower solution branch, using a 2 by 2 mesh, and 4 collocation points per element

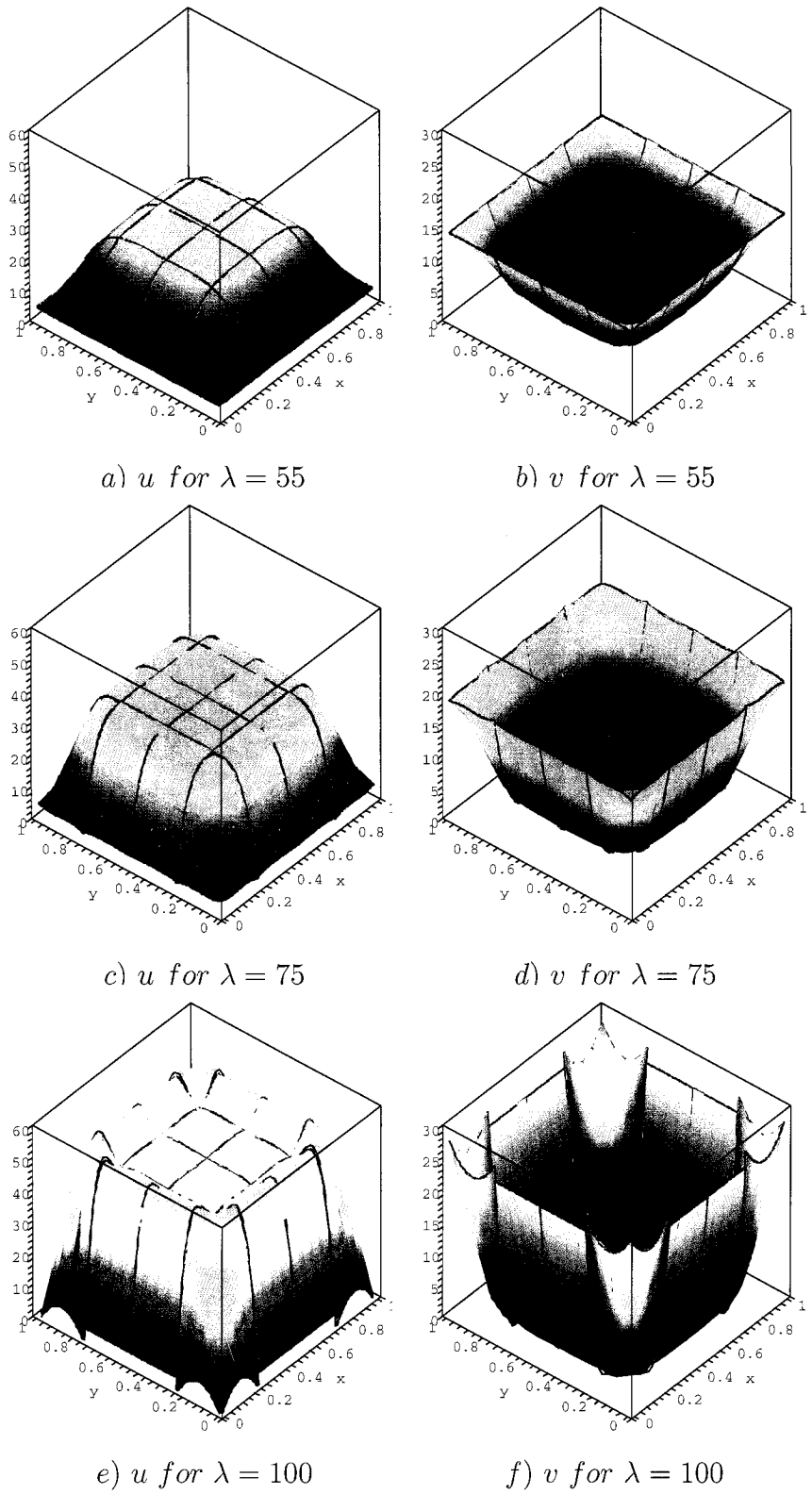


Figure 11.25: Solution of the Brusselator problem, for different values of λ on the upper solution branch, using a 4 by 4 mesh, and 4 collocation points per element

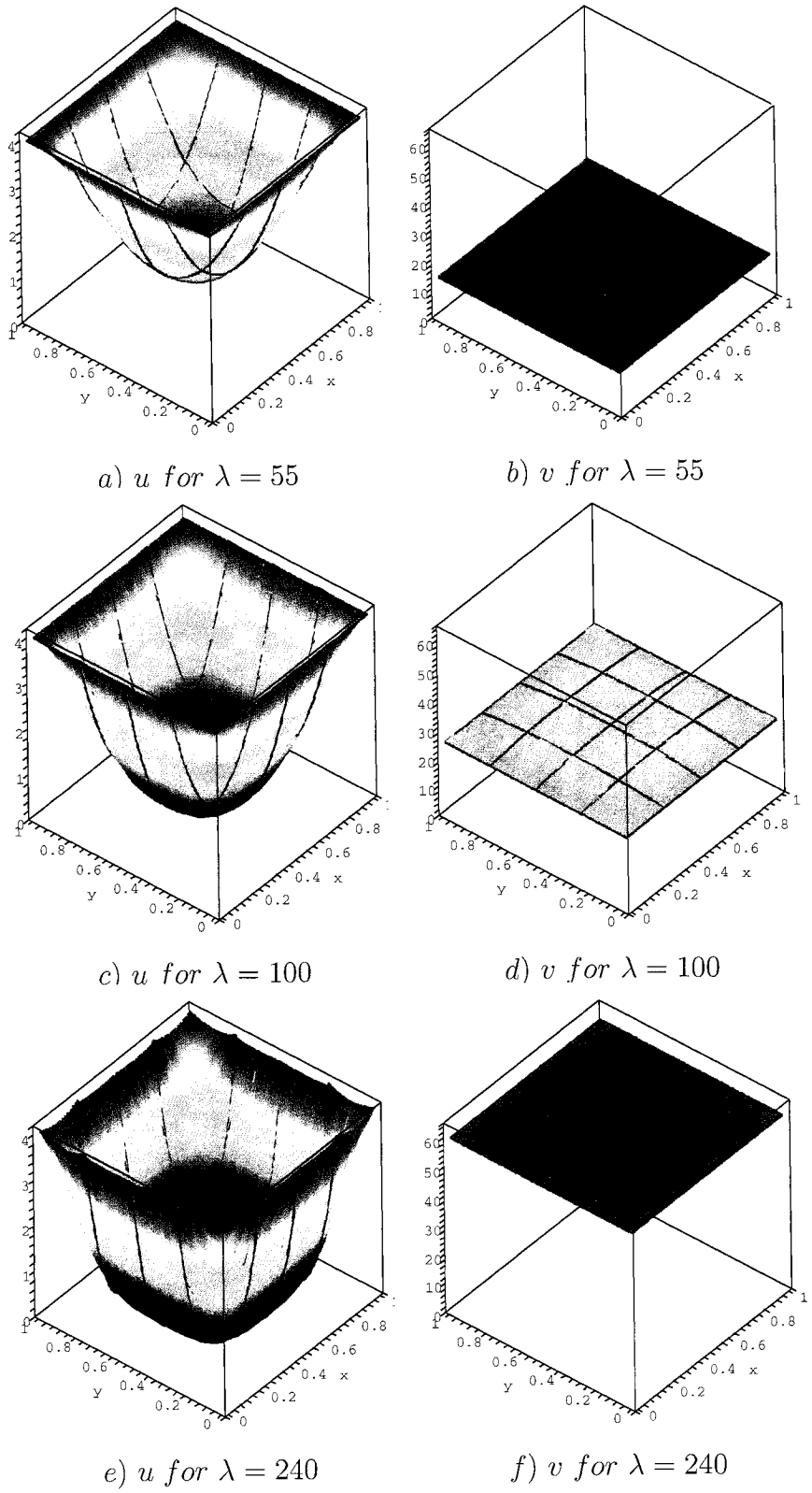


Figure 11.26: Solution of the Brusselator problem, for different values of λ on the lower solution branch, using a 4 by 4 mesh, and 4 collocation points per element

Several authors have used the streamfunction-vorticity formulation of the Navier-Stokes equations for solving two-dimensional incompressible fluid flow problems. The driven cavity flow can be solved using such formulation. One can consult the paper by Spotz [137], who compared different fourth-order compact finite difference discretizations of the streamfunction-vorticity equations for solving the driven cavity problem. He also presented different formulations for handling the no-slip wall boundary conditions.

We have used our prototype software to solve the two-dimensional ($2D$) incompressible Navier-Stokes equations in a square domain with a periodic boundary condition. The relations between the streamfunction (ψ), the vorticity (ζ) and the velocity of a fluid in the steady state $2D$ incompressible fluid flow, can be presented as follows:

$$u = \frac{\partial \psi}{\partial y}, \tag{11.7}$$

$$v = -\frac{\partial \psi}{\partial x}, \tag{11.8}$$

$$\zeta = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}, \tag{11.9}$$

where u and v are components of the velocity of the fluid in x and y directions. Using the above relations, the $2D$ steady state incompressible Navier-Stokes equations can be written as

$$\begin{aligned} &\triangle \psi - \zeta = 0, \\ &\triangle \zeta + R_e(\tfrac{\partial u}{\partial x}\tfrac{\partial v}{\partial y} - \tfrac{\partial u}{\partial y}\tfrac{\partial v}{\partial x}) = 0, \end{aligned} \tag{11.10}$$

where R_e is the Reynolds number. On an unit square domain Ω ($0 \leq x \leq 1,\ 0 \leq y \leq 1$), see Figure 11.27, the following boundary conditions are considered

$$\begin{aligned} &u(x,0) = \tfrac{\partial \psi}{\partial y} = \sin(2\Pi x) && \text{(side 1)}, \\ &v(x,0) = -\tfrac{\partial \psi}{\partial x} = 0 && \text{(side 1)}, \\ &u(x,1) = 0,\ v(x,1) = 0 && \text{(side 3)}, \\ &u(0,y) = u(1,y),\ v(0,y) = v(1,y) && \text{(side 2 and 4)}. \end{aligned} \tag{11.11}$$

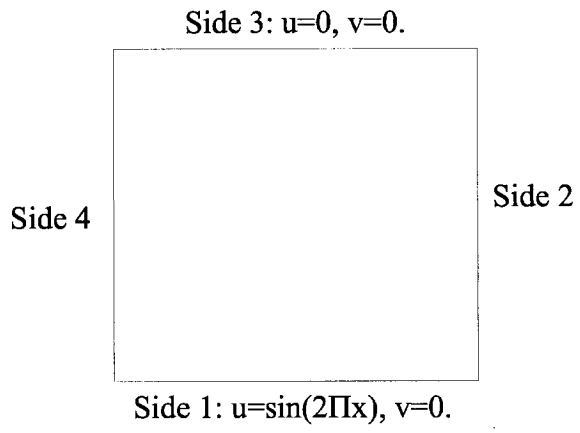


Figure 11.27: A square domain, with periodic boundary condition in the x direction (side 2 and side 4)

The equivalence boundary conditions in terms of ψ and ζ are

$$\begin{aligned}
&\tfrac{\partial\psi(x,0)}{\partial n} = \sin(2\Pi x) && \text{(side 1)},\\
&\psi(x,0) = 0 && \text{(side 1)},\\
&\tfrac{\partial\psi(x,1)}{\partial n} = 0,\ \psi(x,1) = 0 && \text{(side 3)},\\
&\psi(0,y) = \psi(1,y),\ \zeta(0,y) = \zeta(1,y) && \text{(side 2 and 4)},
\end{aligned} \tag{11.12}$$

where n is the outward normal to the boundary. System (11.10) with boundary condition (11.12) is solved using our prototype software. Figure 11.28 presents the bifurcation diagram of the above problem until $R_e = 800$. In this figure, both ψ (the first component) and ζ (the second component) are shown. Figures 11.29 and 11.30 show the variation of each component with respect to R_e, separately.

Figures 11.31, 11.32, 11.33 and 11.34 show the solutions ψ, ζ, u and v for different R_e, using an 8 by 8 mesh and 9 collocation points per element. Figures 11.35, 11.36, 11.37 and 11.38 show the solutions ψ, ζ, u and v for different R_e, using a 16 by 16 mesh, and 9 collocation points per element.

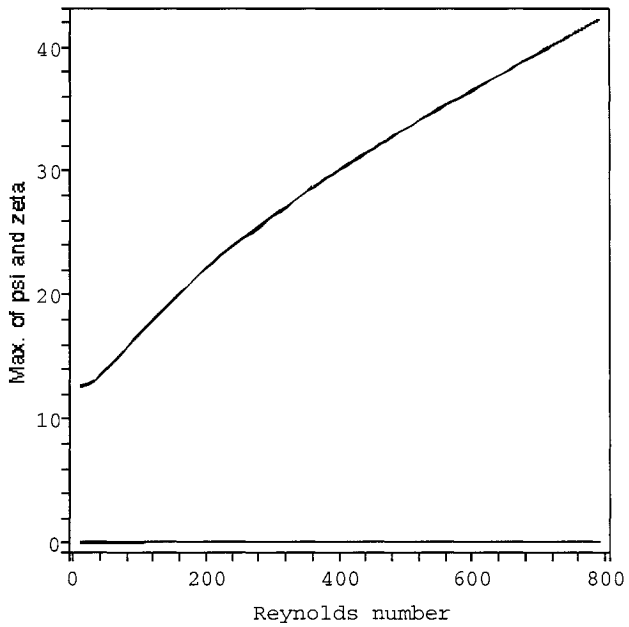


Figure 11.28: Bifurcation diagram of a $2D$ steady state incompressible fluid flow with periodic boundary conditions

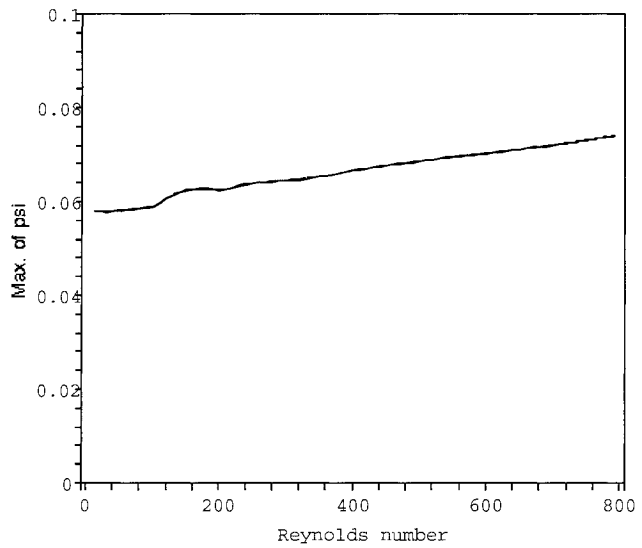


Figure 11.29: Bifurcation diagram showing the first component (ψ) of a $2D$ steady state incompressible fluid flow with periodic boundary conditions

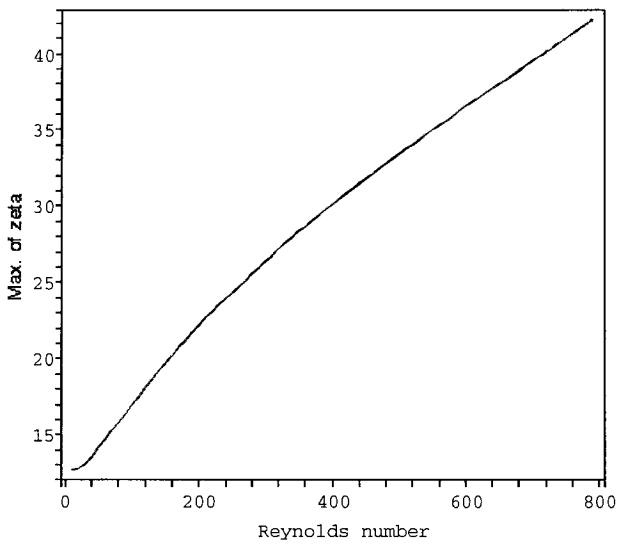


Figure 11.30: Bifurcation diagram showing the second component (ζ) of a $2D$ steady state incompressible fluid flow with periodic boundary conditions

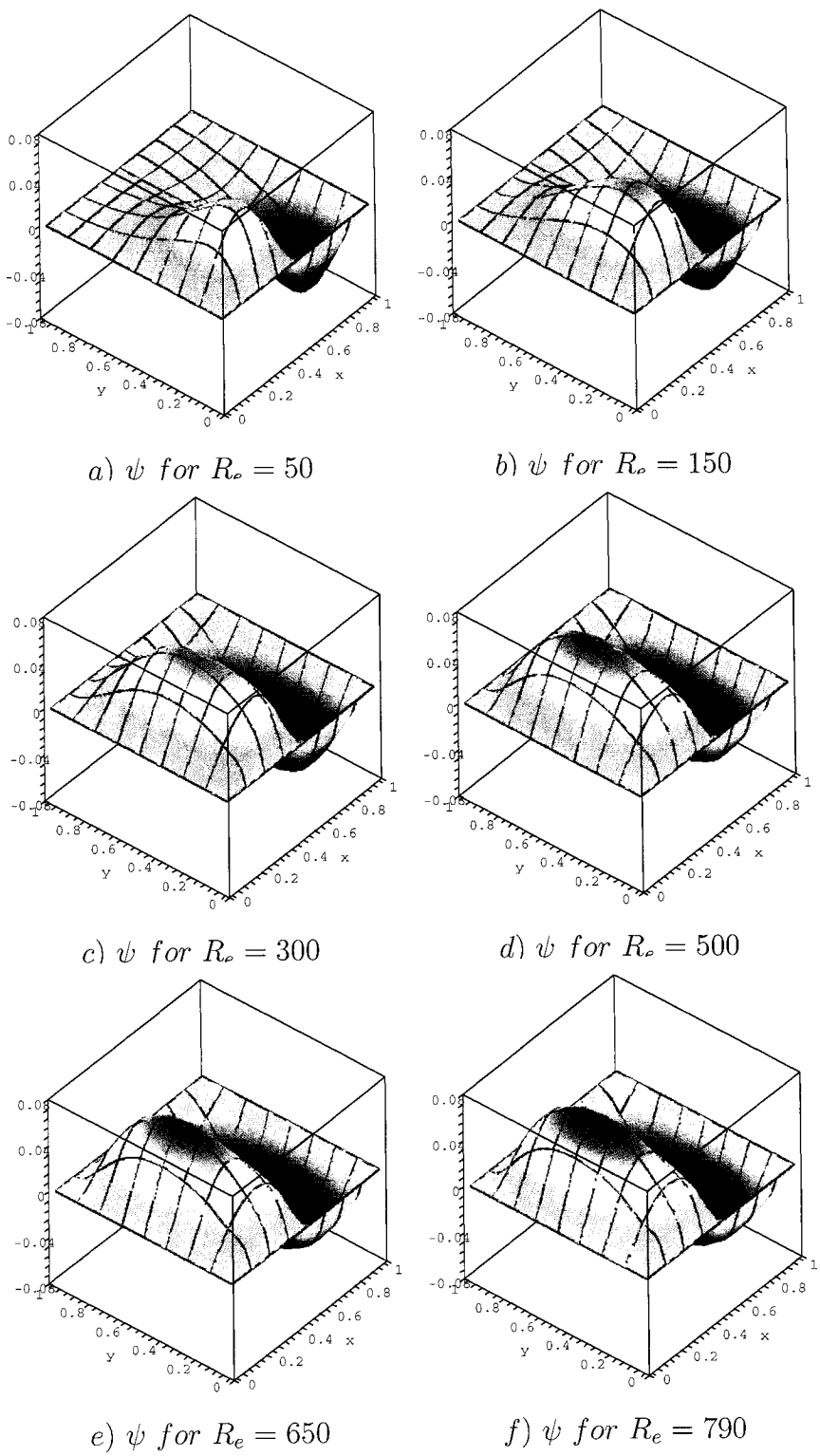
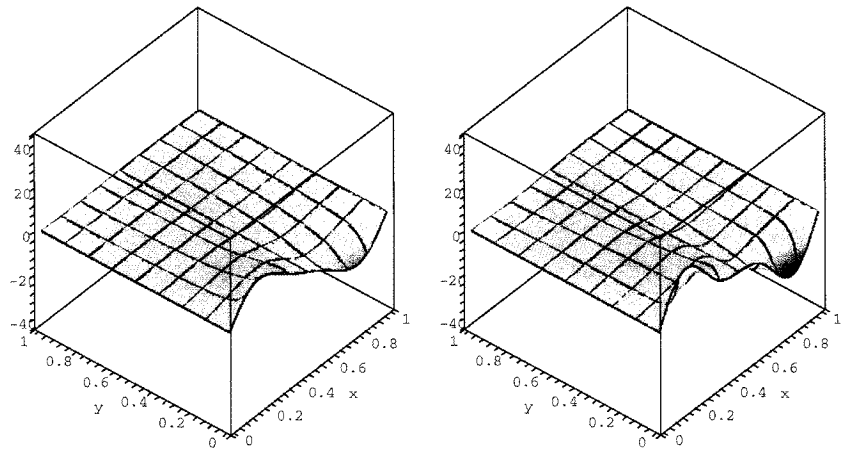
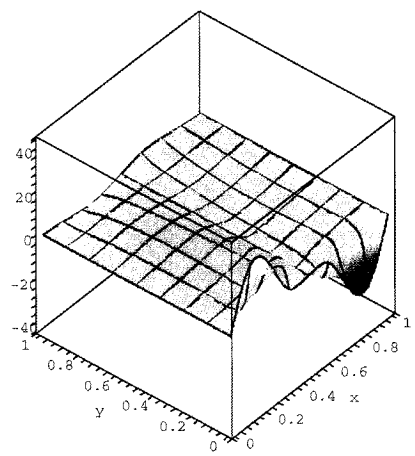


Figure 11.31: The solution ψ of a $2D$ steady state incompressible fluid flow with periodic boundary conditions, for different values of R_e; using an 8 by 8 mesh, and 9 collocation points per element

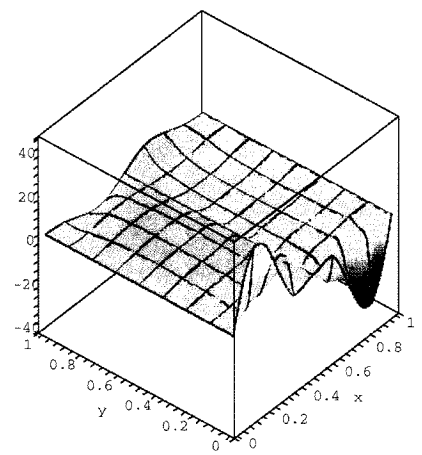


a) ζ *for* $R_e = 50$

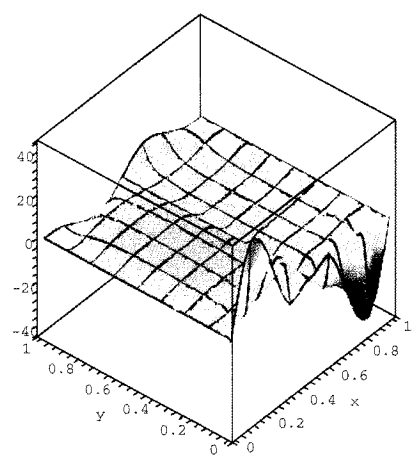
b) ζ *for* $R_e = 150$



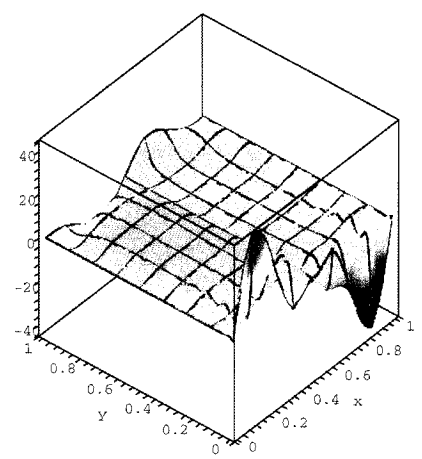
c) ζ *for* $R_e = 300$



d) ζ *for* $R_e = 500$



e) ζ *for* $R_e = 650$



f) ζ *for* $R_e = 790$

Figure 11.32: The solution ζ of a $2D$ steady state incompressible fluid flow with periodic boundary conditions, for different values of R_e; using an 8 by 8 mesh, and 9 collocation points per element

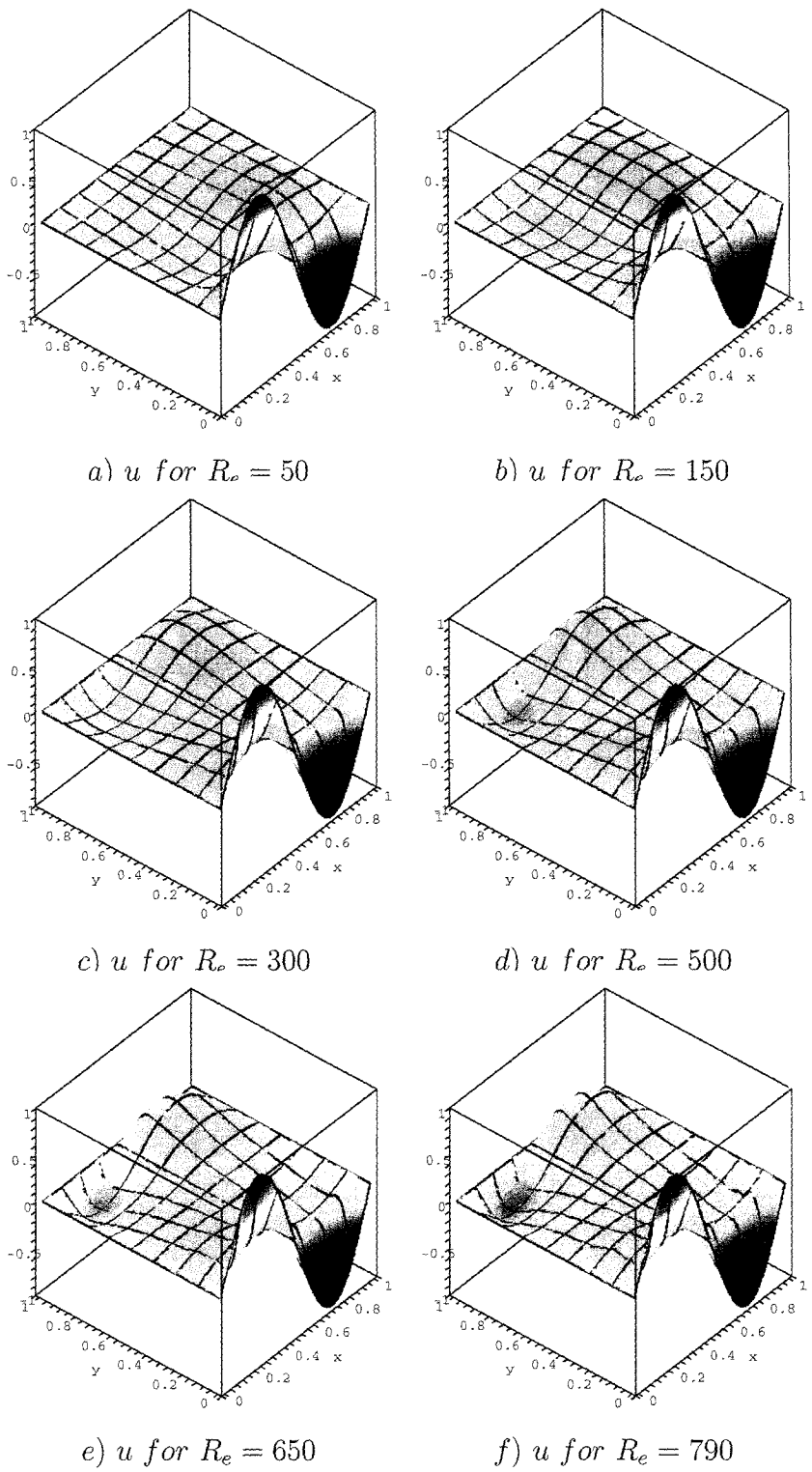
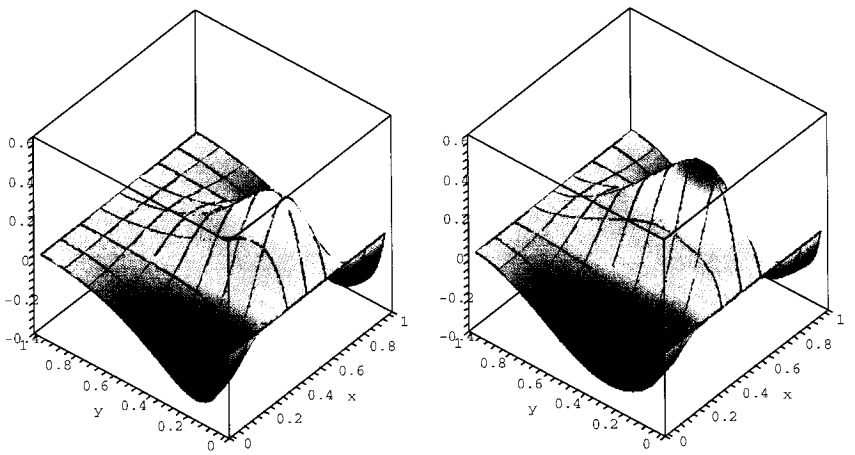
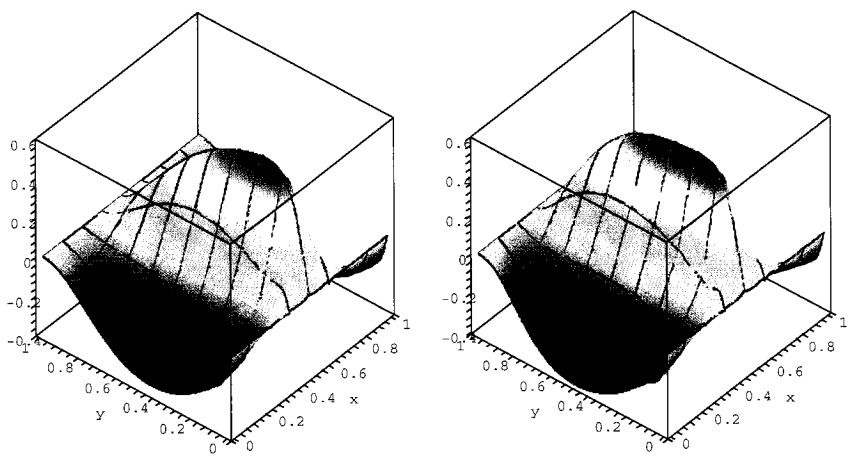


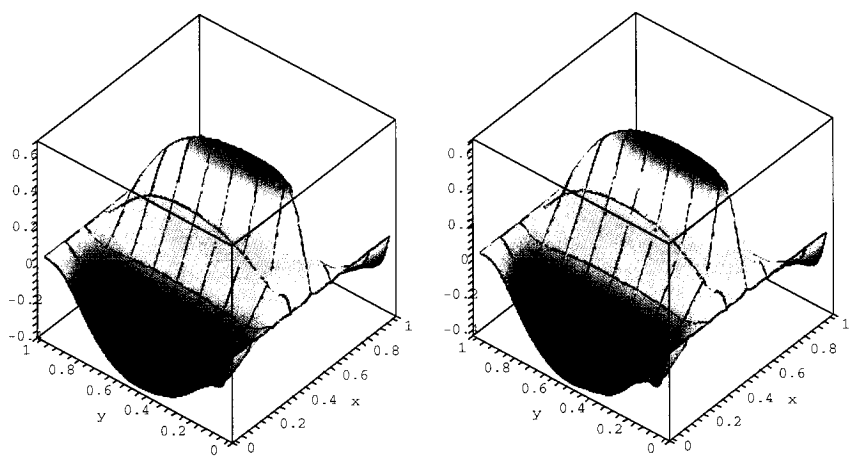
Figure 11.33: The velocity u of a $2D$ steady state incompressible fluid flow with periodic boundary conditions, for different values of R_e; using an 8 by 8 mesh, and 9 collocation points per element



a) v for $R_e = 50$ *b) v for* $R_e = 150$



c) v for $R_e = 300$ *d) v for* $R_e = 500$



e) v for $R_e = 650$ *f) v for* $R_e = 790$

Figure 11.34: The velocity v of a $2D$ steady state incompressible fluid flow with periodic boundary conditions, for different values of R_e; using an 8 by 8 mesh, and 9 collocation points per element

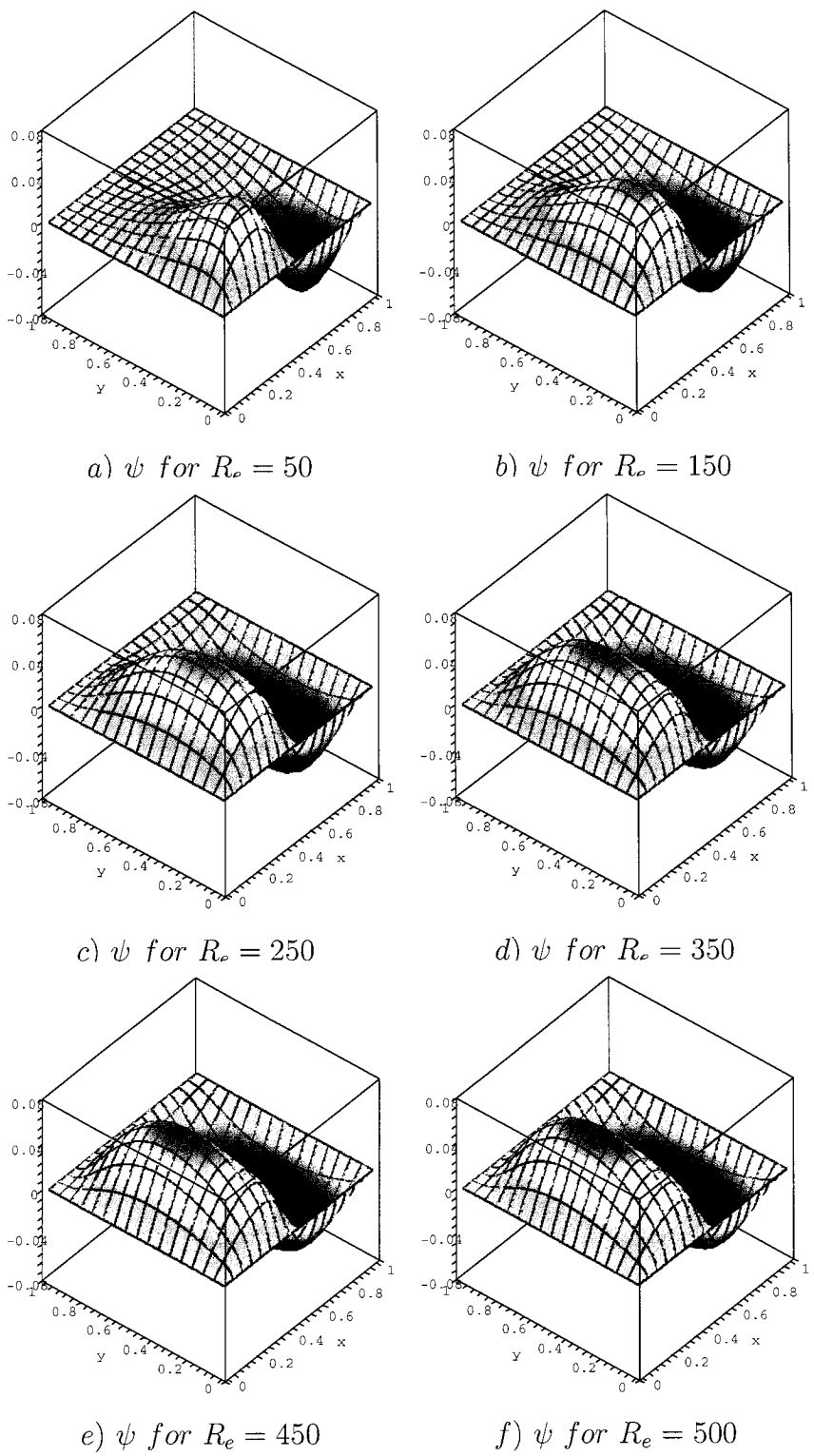
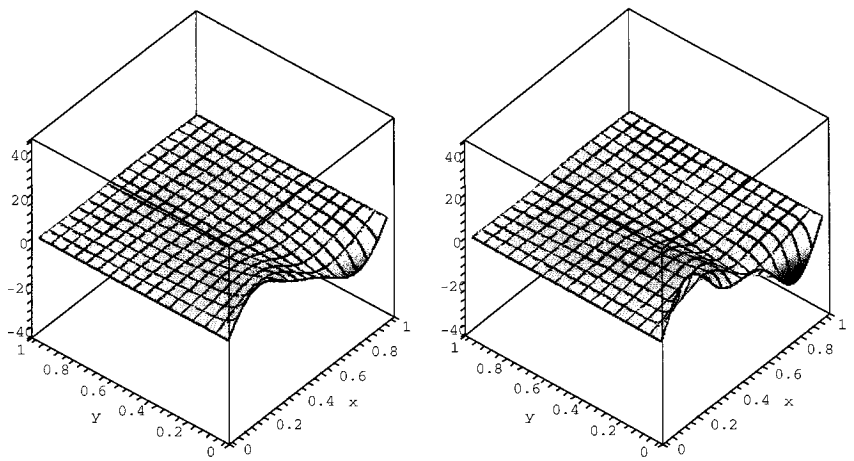
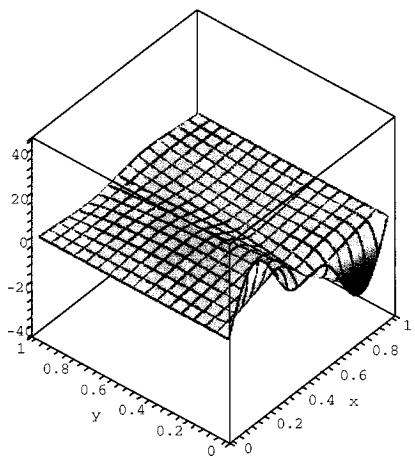


Figure 11.35: The solution ψ of a $2D$ steady state incompressible fluid flow with periodic boundary conditions, for different values of R_e; using a 16 by 16 mesh, and 9 collocation points per element

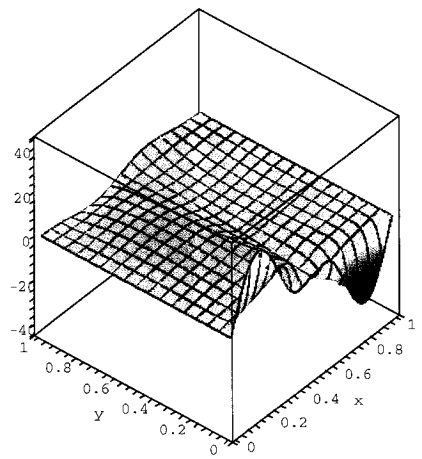


a) ζ for $R_e = 50$

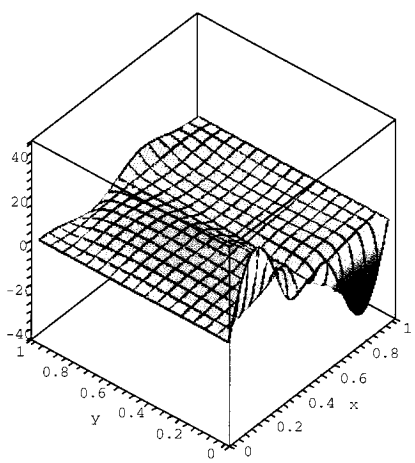
b) ζ for $R_e = 150$



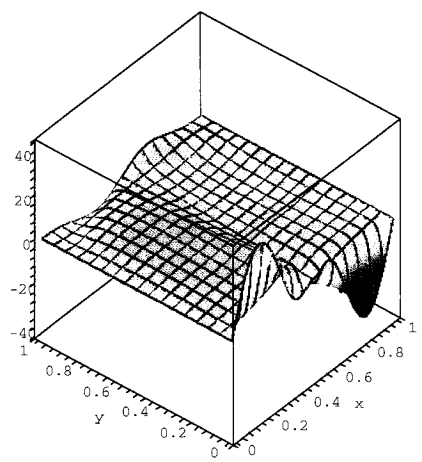
c) ζ for $R_e = 250$



d) ζ for $R_e = 350$

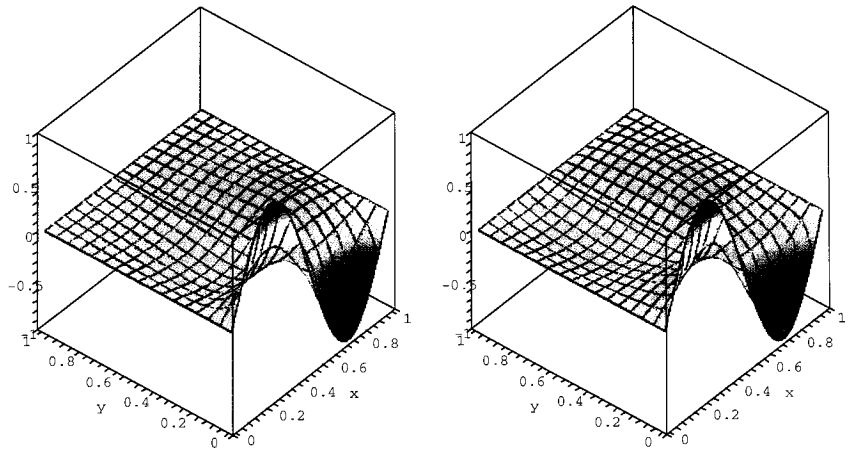


e) ζ for $R_e = 450$

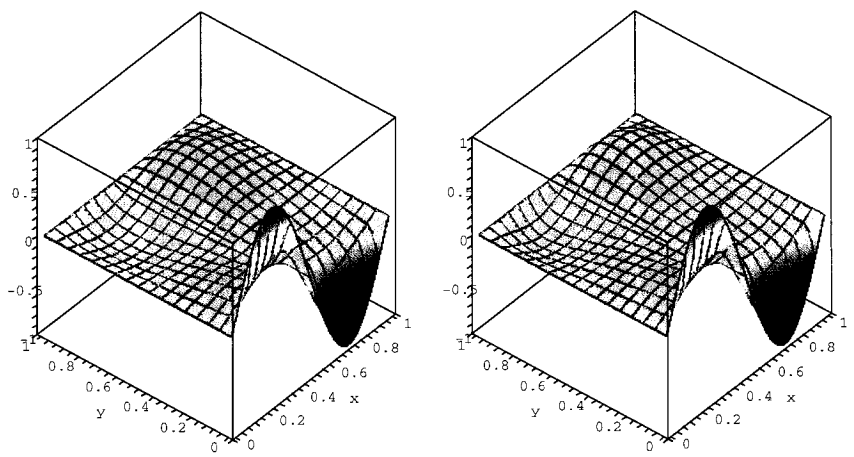


f) ζ for $R_e = 500$

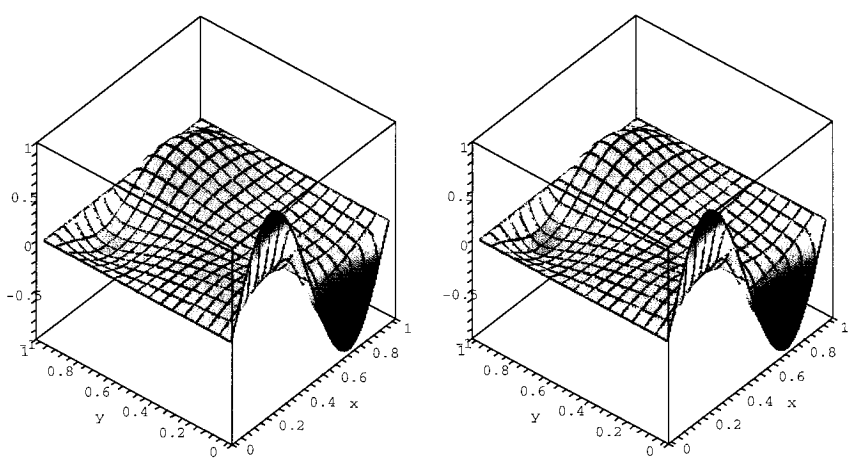
Figure 11.36: The solution ζ of a $2D$ steady state incompressible fluid flow with periodic boundary conditions, for different values of R_e; using a 16 by 16 mesh, and 9 collocation points per element



a) u *for* $R_e = 50$ b) u *for* $R_e = 150$

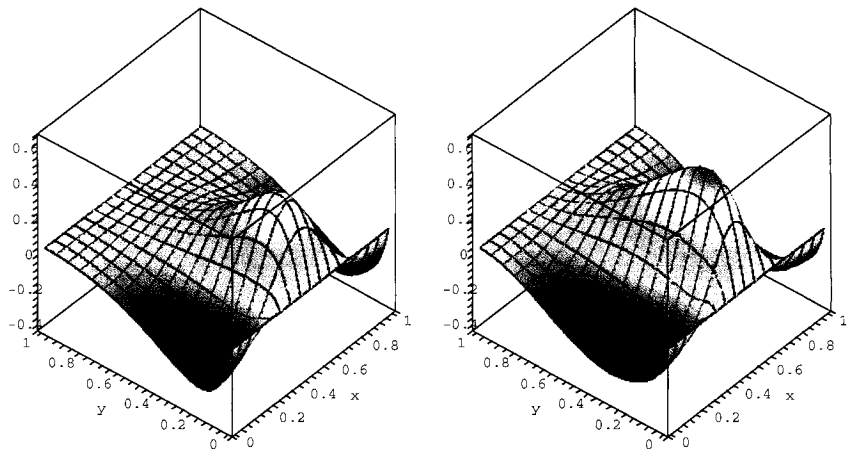


c) u *for* $R_e = 250$ d) u *for* $R_e = 350$

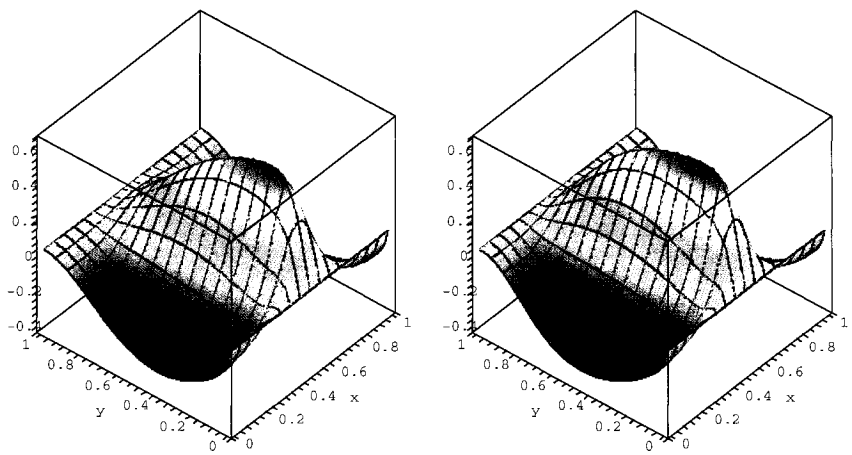


e) u *for* $R_e = 450$ f) u *for* $R_e = 500$

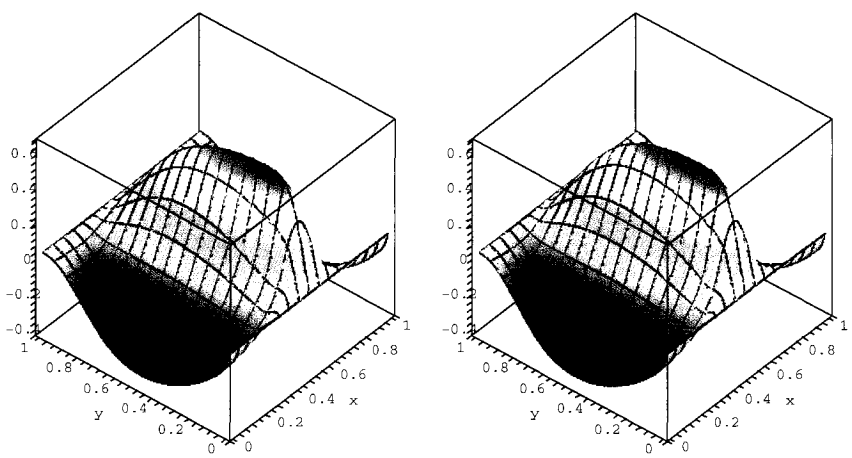
Figure 11.37: The velocity u of a $2D$ steady state incompressible fluid flow with periodic boundary conditions; for different values of R_e; using a 16 by 16 mesh, and 9 collocation points per element



a) v for $R_e = 50$ *b) v for* $R_e = 150$



c) v for $R_e = 250$ *d) v for* $R_e = 350$



e) v for $R_e = 450$ *f) v for* $R_e = 500$

Figure 11.38: The velocity v of a $2D$ steady state incompressible fluid flow with periodic boundary conditions; for different values of R_e; using a 16 by 16 mesh, and 9 collocation points per element

Chapter 12

Conclusion

12.1 Concluding remarks

In this thesis, we have aimed to demonstrate that the piecewise polynomial collocation method is not only a valuable technique for solving boundary value problems in ordinary differential equations, but that it is also an excellent numerical method for solving nonlinear elliptic partial differential equation systems. Various aspects of the finite element piecewise polynomial collocation methods for linear and nonlinear ODEs and PDEs have been discussed in this thesis, and the remarkable unity of the method and the associated solution algorithm, for ODEs and PDEs, were emphasized.

A simple mesh generation procedure was used to produce a suitable mesh. This procedure is based on the recursive subdivision of the domain into two sub-domains, so as to construct a binary tree data structure. This type of domain decomposition is useful in the local mesh refinement process, and also in the robust nested dissection solution algorithm.

For linear ODEs and PDEs, the collocation method was defined equivalently as a type of generalized finite difference method. This finite difference formulation leads

in a very natural way to the nested dissection procedure for solving the discretized equations. For nonlinear ODEs and PDEs, it was explained how the global linearized system is constructed, and how it can be solved using Newton's method and the nested dissection algorithm. Although, the nested dissection method for the case of ODEs does not reduce the computational complexity, as it does for PDEs, it has advantages for bifurcation analysis, for ODEs as well as PDEs.

We have shown that the main characteristics of the discontinuous finite element collocation method are:

- High order of accuracy is possible by increasing the number of collocation and matching points. In fact, the method allows the high order accuracy that is typically required in numerical bifurcation studies.
- For the case of ODEs, the piecewise polynomial solution is globally continuous. However, for the case of PDEs, the piecewise polynomial solutions are not globally continuous. Polynomials corresponding to adjacent elements don't need to match continuously, but their values and normal derivatives match at a discrete set of points on the common boundary.
- The linear systems that arise after discretisation can be solved efficiently by the method of nested dissection.
- Special selection of the local basis functions can reduce the computational cost for uniform and semi-uniform meshes.

For BVPs in linear ODEs we presented a general stability theory, recovering the results of Russell and Shampine [126] and de Boor and Swartz [35], using an alternate method of proof, adapted from Doedel [42], which may also be applicable to the case of PDEs. For certain PDEs, we showed that using 2 by 2 Gauss collocation points, we can reach $\mathcal{O}(h^4)$ convergence, where h is the mesh size. This result is supported by numerical applications.

An important aspect of ODE or PDE analysis is the investigation investigation of solutions as system parameters change. In this thesis, Keller's pseudo-arclength continuation method was used in conjunction with the discontinuous finite element

collocation method for the bifurcation analysis of systems of nonlinear elliptic PDEs. The detection of folds and branch points, the computation of bifurcating direction vectors, and a branch switching procedure were also considered.

We also introduced an alternative formulation of the nested dissection method for constructing the global system from the elementary equations, resolving cases where the original algorithm fails. Specifically, at folds and branch points, one of key matrices in the original nested dissection algorithm may be singular. Furthermore, a modified formulation of the pseudo-arclength equation, in which the step size and the direction vector can be adjusted in each Newton iteration, resolves problems of the original algorithm in the detection of bifurcation points, especially when the boundary conditions depend on a control parameter.

As a bifurcation software package is complex software, object-oriented programming with C++ was used to develop a prototype continuation software package for systems of PDEs. As the object-oriented modelling language to present the implementation algorithms and the prototype software, the UML notation was used. This makes our object-oriented model more understandable, and it also makes further development of the software easier.

For investigating the applicability and efficiency of the method, various applications were considered. These include test applications, *i.e.*, PDEs with known solutions, the Bratu-Gelfand problem, the Brusselator system, and the streamfunction-vorticity formulation of the Navier-Stokes equations for a two-dimensional incompressible fluid flow problem. As boundary conditions, Dirichlet, Neumann, periodic, or a combination of these types were considered. These examples demonstrate the capabilities and the strength of the collocation method with discontinuous elements, and the prototype software, for solving substantial PDEs continuation problems.

12.2 Future work

The suggested further work on the theoretical aspects of the method presented in this thesis, and the improvement of the prototype software can be summarized as follows

- Solving more examples to test the prototype software, to find its possible weaknesses, and to remove them.
- Constructing an acceptable graphical user interface for the software, so that the user be able to enter the PDE systems and the boundaries conditions, to control the continuation process, and to visualize the results easily.
- Integrate the validated software into the AUTO package to provide an AUTO functionality for bifurcation analysis of PDE systems.
- Developing a comprehensive stability and convergence theory for the piecewise polynomial collocation method for nonlinear PDE BVP systems.
- Developing an algorithm to detect and to locate Hopf bifurcations.
- Using triangular elements in place of rectangular elements, and comparing the results.
- Constructing a suitable mesh adaptation strategy, for the rectangular as well as the triangular elements.
- Using more general $2D$ domain, of any shape.
- Applying more general boundary conditions.
- Applying multiple integral constraints.
- developing a parallel implementation of the software.
- Extending this work to the $3D$ domains.

Bibliography

[1] The ITPACK software package. `http://www.ma.utexas.edu/CNA/ITPACK/`, 2003.

[2] M. B. Allen and G. F. Pinder. Collocation simulation of multiphase porous-medium flow. *J. Society of Petroleum Engineers*, pages 135–142, 1983.

[3] E. Allgower and K. Georg. Simplicial and continuation methods for approximating fixed-points and solutions to systems of equations. *SIAM Review*, 22(1):28–85, 1980.

[4] E. L. Allgower and K. Georg. *Numerical Continuation Methods: an Introduction*, volume 13 of *Springer Series In Computational Mathematics*. Springer Verlag, Berlin, Heidelberg, New York, 1990.

[5] E. L. Allgower and K. Georg. *Numerical Continuation and Path Following*, volume 2 of *Springer Series In Computational Mathematics*. Springer Verlag, Berlin, Heidelberg, New York, 1993.

[6] E. L. Allgower and K. Georg. Numerical path following. *P. G. Ciarlet and J. L. Lions, editors, Handbook of Numerical Analysis, North-Holland*, 5:3–207, 1997.

[7] Hassan Aref. Integrable, chaotic, and turbulent vortex motion in two-dimensional flows. *Annual rev. Fluid Mech.*, 15:345–389, 1983.

[8] U. Ascher and G. Bader. Stability of collocation at Gaussian points. *SIAM J. Numer. Anal.*, 23:412–442, 1986.

[9] U. Ascher, J. Christiansen, and R. D. Russell. COLSYS- a collocation code for boundary value problems: in Codes for Boundary-Value Problems in Ordinary

Differential Equations. *B. Childs et al., eds, Lecture Notes in Computer Science 76, Springer-Verlag, New York*, pages 164–185, 1979.

[10] U. M. Ascher, J. Christiansen, and R. D. Russell. A collocation solver for mixed order systems of boundary value problems. *Math. Comp.*, 33:659–679, 1979.

[11] U. M. Ascher, J. Christiansen, and R. D. Russell. Collocation software for boundary value ODEs. *ACM TOMS*, 7(2):209–222, 1981.

[12] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations.* SIAM, Philadelphia, PA, 1995.

[13] U. M. Ascher and R. J. Spiteri. Collocation software for boundary value differential-algebraic equations. *SIAM J. Sci. Comput.*, 15:938–952, 1995.

[14] R. Balart, E. N. Houstis, and T.S. Papatheodorou. On the iteration solution of collocation method equations. *Proceedings of the 10th IMACS Congress*, pages 98–100, 1982.

[15] J. L. Batoz and G. Dhatt. *Modélisation des Structures par Eléments Finis 1, Solides Elastiques.* les presses de l'université Laval, Sainte-Foy, Québec, Canada, 1990.

[16] T. Belytschko and T. L. Geers. Computational methods for fluid-structure interaction problems. *presented at the winter annual meeting of the American Society of Mechanical Engineers*, Atlanta, Georgia, November 27-December 2,1977.

[17] W. J. Beyn and E. J. Doedel. Stability and multiplicity of solutions to discretizations of nonlinear ordinary differential equations. *SIAM J. Sci. Stat. Comput.*, 2(1):107–120, 1981.

[18] B. Bialecki and X. C. Cai. H_1-norm error bounds for piecewise Hermite bicubic orthogonal spline collocation schemes for elliptic boundary value problems. *SIAM J. Numer. Anal.*, 31:1128–1146, 1994.

[19] G. Birkhoff and S. Gulati. Optimal few point discretizations of linear source problems. *SIAM J. Numer. Anal.*, 11:700–728, 1974.

[20] Grady Booch. *Object-Oriented Analysis and Design.* Addison-Wesely, 1994.

[21] D. C. Brabston and H. B. Keller. A numerical method for singular two-point boundary value problems. *SIAM J. Numer. Anal.*, 14:779–791, 1977.

[22] J. C. Cavendish. *Collocation Methods for Elliptic and Parabolic Boundary Value Problems.* PhD thesis, Univ. of Pittsburgh, Pittsburgh, Pa., 1972.

[23] A.R. Champneys, Yu.A. Kuznetsov, and B. Sandstede. A numerical toolbox for homoclinic bifurcation analysis. *Int. J. Bifurcation & Chaos*, 6:867–887, 1996.

[24] C. S. Chien, B. W. Jeng, and C. H. Li. Symmetry reductions and a posteriori finite element error estimators for bifurcation problems. *Int. J. Bifurcation & Chaos*, 15(7), 2005.

[25] C. C. Christara. *Spline Collocation Methods, Software and Architectures for linear Elliptic Boundary Value Problems.* PhD thesis, Purdue University, Lafayette, IN, USA, 1988.

[26] C. C. Christara. Quadratic spline collocation methods for elliptic partial differential equations. *BIT*, 34(1):33–61, 1994.

[27] C. C. Christara. Parallel solvers for spline collocation equations. *Advances in Engineering Software*, 27(1/2):71–89, 1996.

[28] C. C. Christara, E. N. Houstis, and J. R. Rice. A parallel spline collocation - capacitance method for elliptic partial differential equations. *Proceedings of the 1988 International Conference on Supercomputing (ICS88), Saint-Malo, France*, pages 375–384, 1988.

[29] C. C. Christara and Kit Sun Ng. Quadratic spline collocation revisited: Extension to systems of elliptic PDEs. Technical Report 318/2001, Depart. of Computer Science, University of Toronto, Toronto, 2002.

[30] C. C. Christara and Barry Smith. Multigrid and multilevel methods for quadratic spline collocation. *BIT*, 37(4):781–803, 1997.

[31] P. G. Ciarlet. *The Finite Element Method for Elliptic Problems.* North-Holland Publishing Company, New York, 1979.

[32] L. Collatz. *The Numerical Treatment of Differential Equations.* Springer-Verlag, New York, 1966.

[33] A. Conley. *New Plane Shear Flows.* PhD thesis, California Institute of Technology, Pasadena, California, 1994.

[34] A. Constas. Fast Fourier transform solvers for quadratic spline collocation. Master's thesis, Computer Science Dept., Univ. of Toronto, Toronto, 1996.

[35] C. de Boor and B. Swartz. Collocation at Gaussian points. *SIAM J. Numer. Anal.*, 10:583–606, 1973.

[36] G. Dhatt and G. Touzot. *Une Présentation de la Méthode des Eléments Finis.* Maloine, Paris, 1981.

[37] A. Dhooge. *MATCONT: A Matlab Software for Bifurcations of Dynamical Systems.* PhD thesis, Rijksuniversiteit Gent, Belgium, 2005.

[38] A. Dhooge, W. Govaerts, and Yu.A. Kuznetsov. MATCONT: A Matlab package for numerical bifurcation analysis of ODEs. *ACM Transactions on Mathematical Software*, 29(2):141–164, 2003.

[39] David P. Dobkin, Silvio V. F. Levy, William P. Thurston, and Allan R. Wilks. Contour tracing by piecewise linear approximations. *ACM Transactions on Graphics*, 9(4):389–423, 1990.

[40] E. J. Doedel. The construction of finite difference approximations to ordinary differential equations. *SIAM J. Numer. Anal.*, 15(3):450–466, 1978.

[41] E. J. Doedel. Finite difference collocation methods for nonlinear two point boundary value problems. *SIAM J. Numer. Anal.*, 16(2):173–185, 1979.

[42] E. J. Doedel. Some stability theorems for finite difference collocation methods on non-uniform meshes. *BIT*, 20:58–66, 1980.

[43] E. J. Doedel. AUTO, a program for the automatic bifurcation analysis of autonomous systems. *Cong. Numer.*, 30:265–384, 1981.

[44] E. J. Doedel. Nonlinear numerics. *J. Franklin Inst.*, 334B(5/6):1049–1073, 1997.

[45] E. J. Doedel. On the construction of discretizations of elliptic partial differential equations. *Journal of Difference Equations and Applications*, 3:389–416, 1998.

[46] E. J. Doedel. Numerical Analysis. Lecture Notes, COMP361/561, Department of Computer Science, Concordia University, Montréal, Canada, 2000.

[47] E. J. Doedel. Numerical Analysis of Bifurcation Problems. Lecture Notes, Summer School, Technical University of Hamburg, Hamburg, March, 1997.

[48] E. J. Doedel, A.R. Champneys, T.F. Fairgrieve, Yuri A. Kuznetsov, B. Sandstede, and X.J. Wang. AUTO97 : Continuation and bifurcation software for ordinary differential equations. *available by FTP from* `ftp.cs.concordia.ca.` *in directory* `pub/doedel/auto`.

[49] E. J. Doedel, H. B. Keller, and J. P. Kernévez. Numerical analysis and control of bifurcation problems (I) bifurcation in finite dimensions. *Inter. J. of Bifurcation and Chaos*, 1(3):493–520, 1991.

[50] E. J. Doedel, H. B. Keller, and J. P. Kernévez. Numerical analysis and control of bifurcation problems (II) bifurcation in infinite dimensions. *Inter. J. of Bifurcation and Chaos*, 1(4):745–772, 1991.

[51] E. J. Doedel and J.P. Kernévez. AUTO: Software for continuation problems in ordinary differential equations with applications. Technical report, Applied Mathematics, California Institute of Technology, California, 1986.

[52] E. J. Doedel, R.C. Paffenroth, A.R. Champneys, T.F. Fairgrieve, Yu.A. Kuznetsov, B. Sandstede, and X. Wang. AUTO 2000: Continuation and bifurcation software for ordinary differential equations. *available from* `http://auto2000.sourceforge.net/`.

[53] E. J. Doedel and G. W. Reddien. Finite-difference collocation methods for singular boundary value problems. *Approximation Theory III, Academic Press, ISBN: 0-12-171050-5*, pages 349–354, 1980.

[54] E. J. Doedel and G. W. Reddien. Finite difference methods for singular two-point boundary value problems. *SIAM J. Numer. Anal.*, 21(2):300–313, 1984.

[55] E. J. Doedel and H. Sharifi. Collocation methods for continuation problems in nonlinear elliptic PDE systems. *Issue on Continuation Methods in Fluid Mechanics, D. Henry and A. Bergeon, eds., Notes on Numer. Fluid. Mech.*, 74:105–118, 2000.

[56] E. J. Doedel, X. J. Wang, and T. F. Fairgrieve. AUTO94: Software for continuation and bifurcation problems in ordinary differential equations. *Center for Research on Parallel Computing, California Institute of Technology, Pasadena, California 91125*, 1995.

[57] J. Douglas and T. Dupont. A finite element collocation method for quasilinear parabolic equations. *Math. Comp.*, 27:17–28, 1973.

[58] J. Douglas and T. Dupont. Collocation methods for parabolic equations in a single space variable. *Lecture Notes in Mathematics, Springer-Verlag, New York*, 385, 1974.

[59] Y. Dubois-Pèlerin and T. Zimmermann. Object-oriented finite element programming: III. An efficient implementation in C++. *Computer methods in Applied Mechanics and Engineering*, 108:165–183, 1993.

[60] W. R. Dyksen. Tensor product generalized ADI methods for separable elliptic problems. *SIAM J. Numer. Anal.*, 24:59–76, 1987.

[61] W. R. Dyksen and J. R. Rice. A new ordering scheme for the Hermite bicubic collocation equations. *In Elliptic Problem Solvers III*, Academic Press, New York:467–480, 1984.

[62] K. Engelborghs. DDE-BIFTOOL: a Matlab package for bifurcation analysis of delay differential equations. Technical Report TW-305, Department of Computer Science, K.U.Leuven, Leuven, Belgium, 2000.

[63] K. Engelborghs and E. J. Doedel. Convergence of a boundary value difference equation for computing periodic solutions of neutral delay differential equations. *J. Difference Equations and Applications*, 7:927–940, 2001.

[64] K. Engelborghs and E. J. Doedel. Stability of piecewise polynomial collocation for computing periodic solutions of delay differential equations. *Numerische Mathematik*, 91(4):627–648, 2002.

[65] K. Engelborghs, T. Luzyanina, K.J. in 't Hout, and D. Roose. Collocation methods for the computation of periodic solutions of delay differential equations. *SIAM J. Scientific Computing*, 22(5):1593–1609, 1999.

[66] K. Engelborghs and D. Roose. Numerical computation of stability and detection of Hopf bifurcations of steady state solutions of delay differential equations. *Advances in Computational Mathematics*, 10(3-4):271–289, 1999.

[67] J.A. Essers. *Computational Methods for Turbulent, Transonic and Viscous Flows.* Hemisphere Pub, August 1983.

[68] G. Fairweather. Spline collocation methods for a class of hyperbolic partial integro-differential equations. *SIAM J. Numer. Anal.*, 31:444–460, 1994.

[69] A. I. Fedoseyev, M. J. Friedman, and E. J. Kansa. Continuation for nonlinear elliptic partial differential equations discretized by the multiquadric method. *Int. J. Bifurcation & Chaos*, 10(2):481–492, 2000.

[70] B. W. Forde and al. Object-oriented finite element analysis. *Computers & Structures*, 34(3):355–374, 1990.

[71] C. B. Garcia and W. I. Zangwill. *Pathways to Solutions, Fixed Points, and Equilibria.* Prentice-Hall, Englewood Cliffs, NJ., 1981.

[72] C. W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall, Englewood Cliffs, NJ, 1971.

[73] B. Goldlücke. Nichtkonforme finite Elemente und Doedel-Kollokation für elliptische Differentialgleichungen. Master's thesis, Diplomarbeit Fachbereich Mathematik und Informatik Philipps-Universität, Marburg, November 2001.

[74] F. J. Gould and J. W. Tolle. *Complementary Pivoting on a Pseudomanifold Structure with Applications in the Decision Sciences*, volume 2 of *Sigma Series in Applied Mathematics*. Heldermann Verlag, Berlin, 1983.

[75] Ian Graham. *Object-Oriented Methods, Principles & Practice, Third Edition*. Addison-Wesley, 2001.

[76] R. D. Grigorieff. Die Konvergenz des Rand und Eigenwert Problems Linearer Gewöhnlicher Differenzengleichungen. *Numer. Math.*, pages 15–48, 1970.

[77] M. E. Henderson. Continuation methods. *available from the Applied Mathematics Group in the Mathematical Sciences Department at IBM's T.J. Watson Research Center*, `http://www.research.ibm.com/people/h/henderson/Continuation/ContinuationMethods.html`, 2002.

[78] Michael E. Henderson. Multiple parameter continuation: Computing implicitly defined k-manifolds. *Int. J. Bifurcation & Chaos*, 12(3):451–476, 2002.

[79] F. R. De Hoog and R. Weiss. Collocation methods for singular boundary value problems. *SIAM J. Numer. Anal.*, 15:198–217, 1978.

[80] E. N. Houstis, C. C. Christara, and J. R. Rice. Quadratic spline collocation methods for two point boundary value problems. *International Journal for Numerical Methods in Engineering*, 26(4):935–953, 1988.

[81] W. Huang and D. Sloan. A new pseudospectral method with upwind features. *IMA J. Numer. Anal.*, 13:413–430, 1993.

[82] T. Ito. *A Collocation Methods for Boundary Value problems Using Spline Functions*. PhD thesis, Brown Univ., Providence, R.I., 1972.

[83] H. B. Keller. *Numerical Methods for Two Point Boundary Value Problems*. Blaisdell, London, 1968.

[84] H. B. Keller. A new difference scheme for parabolic problems. In *Numerical solutions of partial differential equations, II (Hubbard, B. ed.)*, pages 327–350, New York, 1971. Academic Press.

[85] H. B. Keller. Numerical solution of two point boundary value problems. *Regional Conference Series in Applied Mathematics*, Society for Industrial and Applied Mathematics, Philadelphia(24), 1976.

[86] H. B. Keller. Numerical solution of bifurcation and nonlinear eigenvalue problems. *In: Application of Bifurcation Theory, ed. P. H Rabinowitz, Academic Press*, pages 359–384, 1977.

[87] H. B. Keller. *Lectures on Numerical Methods in Bifurcation Problems.* Springer Verlag, Berlin Heidelberg, New York, 1987.

[88] H. B. Keller and V. Pereyra. Difference methods and deferred corrections for ordinary boundary value problems. *SIAM J. Numer. Anal.*, 16(2):241–259, 1979.

[89] Kelvin. Vibrations of a columnar vortex. *Proc. of the Royal Society, Edinburgh*, 1880.

[90] X. A. Kong and D. P. Chen. An object-oriented design of FEM programs. *Computers & Structures*, 57(1):167–166, 1995.

[91] H. O. Kreiss. Difference approximations for boundary and eigenvalue problems for ordinary differential equations. *Math. Comput.*, 26:605–624, 1972.

[92] Yu. A. Kuznetsov, V. V. Levitin, and A.R. Skovoroda. Continuation of stationary solutions to evolution problems in CONTENT. Report AM-R9611, Centrum voor Wiskunde en Informatica, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands, 1996.

[93] Yu. A. Kuznetsov and V.V. Levitin. CONTENT: A multiplatform environment for continuation and bifurcation analysis of dynamical systems. *Centrum voor Wiskunde en Informatica, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands*, 1997.

[94] C. Lanczos. Trigonometric interpolation of empirical and analytical functions. *J. Math. Phys.*, 17:123–199, 1938.

[95] L. P. Lebedev, I. I. Vorovich, and G. M. L. Gladwell. *Functional analysis; Applications in mechanics and inverse problems.* Springer, Boston Kluwer Academic Publishers, New York, 2 edition, 2002.

[96] Randall J. LeVeque. Finite difference methods for differential equations. *Lecture note of the course AMath 585-6*, University of Washington, 2001.

[97] K. Lust, D. Roose, A. Spence, and A.R. Champneys. An adaptive Newton-Picard algorithm with subspace iteration for computing periodic solutions. *SIAM Journal on Scientific Computing*, 19(4):1188–1209, 1998.

[98] D. Mahmood and M. R. Osborne. Collocation for BVP's: compact and noncompact schemes. *In D. Stewart, H. Gardner and D. Singleton, editors, Computational Techniques and Applications: CTAC93, World Scientific*, pages 354–361, 1994.

[99] J. E. Marsden and M. J. Hoffman. *Elementary Classical Analysis.* W.H. Freeman & Company, New York, 2 edition, 1993.

[100] Mathworks. The Finite Element Method (Partial Differential Equation Toolbox). `http://www.mathworks.com/access/helpdesk/help/toolbox/pde/4fem2.shtml`, 2003.

[101] P. Menétry and T. Zimmermann. Object-oriented nonlinear finite element analysis: application to $j2$ plasticity. *Computer & Structures*, 49(5):767–777, 1993.

[102] R. Meyer-Spasche and H. B. Keller. Numerical study of Taylor vortex flows between rotating cylinders. *Applied Mathematics, California Institute of Technology, Pasadena, California 91125*, July 1978.

[103] R. Meyer-Spasche and M. Wagner. Steady axisymmetric Taylor vortex flows with free stagnation points of the poloidal flow. *Inter. Series of Numerical Mathematics*, 79, 1987.

[104] Pierre-Alain Muller. *Instant UML.* Wrox Press, 1997.

[105] T. Mullin. *The Nature of Chaos.* Oxford Science Publication, 1993.

[106] Kit Sun Ng. Quadratic Spline Collocation Methods for Systems of Elliptic PDEs. Master's thesis, Computer Science Dept., Univ. of Toronto, Toronto, 2000.

[107] Site of ArgoUML. ArgoUML software. `http://argouml.tigris.org/index.html`, 2003.

[108] Site of numerical methods. com. Finite difference method. `http://www.numerical-methods.com/`, 2003.

[109] Rational Rose Group of the IBM. Rational rose software. `http://www.rational.com/products/rose/index.jsp`, 2003.

[110] OMG. *OMG Unified Modeling Language Specification, Version 1.5.* Object Management Group, Inc., http://www.omg.org, March 2003.

[111] M. R. Osborne. A method for finite difference approximation to ordinary differential equations. *Computer J.*, 7:58–65, 1964.

[112] M. R. Osborne. Minimising truncation error in finite difference approximation to ordinary differential equations. *Math. Comput.*, 11:133–145, 1967.

[113] M. R. Osborne. Collocation, difference equations, and stitched function representations. *Topics in Numerical Analysis II, (ed. J. H. Miller), Academic Press, New York*, 1974.

[114] P. Percell and M. F. Wheeler. A C^1 finite element collocation method for elliptic equations. *SIAM J. Numer. Anal.*, 17(5):605–622, 1980.

[115] P. M. Prenter and R. D. Russell. Orthogonal collocation for partial differential equations. *SIAM J. Numer. Anal.*, 13:923–939, 1976.

[116] C. Quézel, Mn. Fafard, and M. Fortin. Object-oriented finite element analysis. *GIREF, Université Laval*, 1996.

[117] G. W. Reddien. Projection methods and singular two-point boundary value problems. *Numer. Math.*, 21:193–203, 1973.

[118] G. W. Reddien and L. L. Schumaker. On a collocation method for singular two-point boundary value problems. *Numer. Math.*, 25:472–432, 1976.

[119] W. C. Rheinboldt. *Numerical Analysis of Parametrized Nonlinear Equations.* J. Wiley, New York, New York, 1986.

[120] J. R. Rice and R. F. Boisvert. *Solving Elliptic Problems Using Ellpack.* Springer-Verlag, New York, 1985.

[121] C. Ringhofer. On collocation schemes for quasilinear singularly perturbed boundary value problems. *Springer-Verlag, New York*, 21:864–882, 1984.

[122] D. Roose, K. Lust, A.R. Champneys, and A. Spence. A Newton-Picard shooting method for computing periodic solutions of large-scale dynamical systems. *Chaos, Solitons and Fractals*, 5(10):1913–1925, 1995.

[123] Jim Rumbaugh, Mike Blana, Bill Premerlani, Fred Eddy, and Bill Lorensen. *Object-Oriented Modeling and Design.* Prentice Hall, 1991.

[124] R. D. Russell. Collocation for systems of boundary value problems. *Numer. Math.*, 23:119–133, 1974.

[125] R. D. Russell and J. Christiansen. Adaptive mesh selection strategies for solving boundary value problems. *SIAM J. Numer. Anal.*, 15:59–80, 1978.

[126] R. D. Russell and L. F. Shampine. A collocation method for boundary value problems. *SIAM J. Numer. Anal.*, 19:1–28, 1972.

[127] R. D. Russell and L. F. Shampine. Numerical methods for singular two-point boundary value problems. *SIAM J. Numer. Anal.*, 12:13–36, 1975.

[128] R. D. Russell and W. Sun. Spline collocation differentiation matrices. *SIAM J. Numer. Anal.*, 34:2274–2287, 1997.

[129] R. D. Russell and J. M. Varah. A comparison of global methods for two-point boundary value problems. *Math. Comp.*, 29:1–13, 1975.

[130] A. G. Salinger, N. M. Bou-Rabee, R. P. Pawlowski, E. D. Wilkes, E. A. Burroughs, R. B. Lehoucq, and L. A. Romero. LOCA: Library of continuation algorithms: Theory and implementation manual. SAND REPORT SAND2002-0396, Sandia National Laboratories, 2002.

[131] S. P. Scholz. Elements of an object-oriented FEM++ program in C++. *Computers & Structures*, 43(3):517–529, 1992.

[132] L. J. Segerlind. *Applied Finite Element Analysis.* Wiley, New York, 1976.

[133] R. Seydel. *From Equilibrium to Chaos. Practical Bifurcation and Stability Analysis.* Elsevier, New York, 1988.

[134] H. Sharifi. Méthode des elément finis de frontière utilisant la programmation orientée objet. Master's thesis, Département d'informatique, Université Laval, Ste-Foy, Québec, Canada, September 1997.

[135] H. Sharifi and A. Gakwaya. Object-oriented modelling of field boundary element method in nonlinear solid mechanics with applications. In A. P. S. Selvadurai, C. L. Tan, and M. H. Aliabadi, editors, *Advances in Boundary Element Techniques VI*, pages 317–326. EC, Ltd, UK, July 2005.

[136] J. Smoller. *Shock Waves and Reaction-Diffusion Equations.* Springer-Verlag, 1994.

[137] W. F. Spotz. Accuracy and performance of numerical wall boundary conditions for steady, $2d$, incompressible streamfunction vorticity. *Int. J. Numer. Meth. Fluids*, 28:737–757, 1998.

[138] W. Sun. A high order direct method for solving Poisson's equations in a disc. *Numer. Math.*, 70:501–506, 1995.

[139] W. Sun. Iterative algorithm for orthogonal spline collocation linear system. *SIAM. J. Sci. Comput.*, 16:720–737, 1995.

[140] W. Sun. Cyclic reduction algorithms for solving collocation systems. *Intern. J. Computer Math.*, 61:293–305, 1996.

[141] W. Sun. Fast algorithms for solving high-order spline collocation systems. *Numer. Math.*, 81:143–160, 1998.

[142] W. Sun. Hermite cubic spline collocation methods with upwind features. *ANZIAM J.*, 42:C1379–C1397, 2000.

[143] W. Sun and J. Wu. personal communication. 2004.

[144] W. Sun and N. G. Zamani. A fast algorithm for solving the tensor product collocation equations. *J. of the Franklin Institute*, 326(2):295–307, 1989.

[145] B. K. Swartz. The construction and comparison of finite difference analogs of some finite element schemes. Report, Los Alamos Scientific Laboratory, Los Alamos, NM., 1974.

[146] G. I. Taylor. Stability of a viscous liquid contained between two rotating cylinders. *Phil. Trans. R. Soc. A*, 233:289–343, 1923.

[147] R. Temam. *Infinite-Dimensional Dynamical Systems in Mechanics and Physics.* Springer-Verlag, 1993.

[148] M. J. Todd. The computation of fixed points and applications. *Lecture Notes in Economics and Mathematical Systems, Springer Verlag, Berlin, Heidelberg, New York*, 124, 1976.

[149] R. S. Varga. *Matrix Iterative Analysis.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1962.

[150] G. H. Vatistas. A note on liquid vortex sloshing and Kelvin's equilibria. *J. Fluid Mech.*, 217:241–248, 1990.

[151] Bruce E. Wampler. *The Essence of Object-Oriented Programming with Java and UML.* Addison-Wesely, 2002.

[152] J. Wang. *On the Wave Activity within Vortex Cores.* PhD thesis, Department of Mechanical Eng., Concordia University, Montreal, Québec, Canada, 1995.

[153] T. Zimmermann Y. Dubois-Pèlerin and P. Bomme. Object-oriented finite element programming: II. a prototype program in Smalltalk. *Computer methods in Applied Mechanics and Engineering*, 98:361–397, 1992.

[154] G. W. Zeglinkski, P. S. Han, and P. Aitchison. Object-oriented matrix classes for use in a finite element code using C++. *International Journal for Numerical Methods in Engineering*, 37:3921–3937, 1994.

[155] O. C. Zienkiewicz. *The Finite Element Method.* McGraw Hill, New York, 1977.