

DISSERTATION

Numerical Solutions of Nonlinear Systems Derived From
Semilinear Elliptic Equations

Submitted by
Stefan-Gicu Cruceanu
Department of Mathematics

In partial fulfillment of the requirements
For the Degree of Doctor of Philosophy
Colorado State University
Fort Collins, Colorado
Spring 2007

UMI Number: 3266385

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI[®]

UMI Microform 3266385

Copyright 2007 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Copyright by Stefan-Gicu Cruceanu 2007

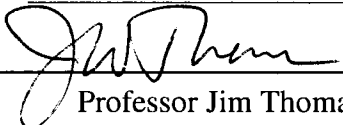
All Rights Reserved

COLORADO STATE UNIVERSITY

March 6, 2007

WE HEREBY RECOMMEND THAT THE **DISSERTATION** PREPARED UNDER OUR SUPERVISION BY **STEFAN-GICU CRUCEANU** ENTITLED **NUMERICAL SOLUTIONS OF NONLINEAR SYSTEMS DERIVED FROM SEMILINEAR ELLIPTIC EQUATIONS** BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

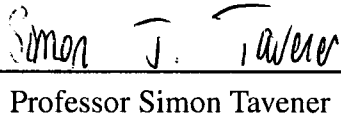
Committee on Graduate Work



Professor Jim Thomas

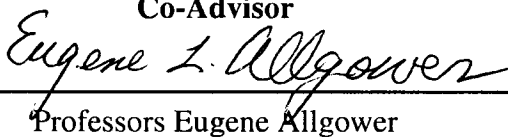


Professor Martin Gelfand



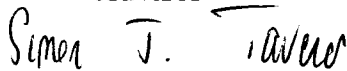
Professor Simon Tavener

Co-Advisor



Professors Eugene Allgower

Advisor



Professor Simon Tavener

Professor Simon Tavener
Department Head/Director

ABSTRACT OF DISSERTATION

Numerical Solutions of Nonlinear Systems Derived From Semilinear Elliptic Equations

The existence and the number of solutions for N -dimensional nonlinear boundary value problems has been studied from a theoretical point of view, but there is no general result that states how many solutions such a problem has or even to determine the existence of a solution. Numerical approximation of all solutions (complex and real) of systems of polynomials can be performed using numerical continuation methods. In this thesis, we adapt numerical continuation methods to compute all solutions of finite difference discretizations of boundary value problems in 2-dimensions involving the Laplacian. Using a homotopy deformation, new solutions on finer meshes are obtained from solutions on coarser meshes. The issue that we have to deal with is that the number of the solutions of the complex polynomial systems grows with the number of mesh points of the discretization. Hence, the need of some filters becomes necessary in this process.

We remark that in May 2005, E. Allgower, D. Bates, A. Sommease, and C. Wampler used in [1] a similar strategy for finding all the solutions of two-point boundary value problems in 1-dimension with polynomial nonlinearities on the right hand side. Using exclusion algorithms, we were able to handle general nonlinearities.

When tracking solutions sets of complex polynomial systems an issue of bifurcation or near bifurcation of paths arises. One remedy for this is to use the gamma-trick introduced by Sommease and Wampler in [2]. In this thesis we show that bifurcations necessarily occur at turning points of paths and we use this fact to numerically handle the bifurcation, when mappings are analytic.

Stefan-Gicu Cruceanu
Department of Mathematics
Colorado State University
Fort Collins, Colorado 80523
Spring 2007

ACKNOWLEDGEMENTS

This thesis wouldn't have been created if I hadn't had the wonderful opportunity to work with professors of a great intellectual formation, people who had a positive and constructive impact on my career.

I am deeply grateful to my advisors, Professors Eugene L. Allgower and Simon Tavener, for their ideas, guidance, patience and encouragement throughout my course work and research at Colorado State University. Eugene is the one who introduced me to this relatively new direction in Mathematics, which has been elaborated in our paper. It attracted my attention and interest from the very beginning, and due to their experience and constructive suggestions I was able to pass different obstacles I have met along my research. They helped me understand, interpret and overcome when necessary some of the numerical difficulties I have struggled with and facilitated my walking on this path which we all have built. I highly appreciate the countless hours they have spent reviewing, commenting and improving this manuscript.

A special 'thank you' goes to Professor Jim Thomas; after going through [3] and having some interesting talks with him, I was able to build the moving mesh algorithm for a rectangular domain, as well as the discretizations for different derivatives that take place on such meshes. I also want to thank Jim for his wonderful ideas that helped me a lot in my two years (2003-2005) of soccer coaching for Arsenal White Team (Fort Collins).

My sincere appreciation goes to Professor Martin Gelfand also, for having served on my committee and providing with his help.

I would like to express my gratitude to Professor Kurt Georg who was a co-advisor for me before he passed away suddenly on February 14, 2003. He was a numerical analyst and a leading exponent of numerical continuation methods. Our discussions enlarged my

perceptions about this field. In 2006, I had to build my own toolbox of programs for the exclusion algorithms and fortunately, I was able to get the same numerical results as in [4]. I used this toolbox in our research to generalize our homotopy continuation for the case of a nonpolynomial nonlinearity (Chapter IV).

A big thank you to Professors Iuliana Oprea and Kelly McArthur who helped me in the early stages of my becoming a better mathematician. The classes they have taught me and their guidance oriented me toward applied mathematics.

Irma Woollen deserves a special note of thanks for being a morale pillar to me in my early years at Colorado State University.

I am deeply grateful to my parents for inculcating a learning spirit into me during my childhood, for their love, help and encouragement throughout my whole life. My brother and they mean a lot to me, and therefore, going away from them for this period of study was a sacrifice from both sides. But it was all worth it...

Last, but not least, I would like to thank my wife, Ana, for her patience, encouragement and support throughout my graduate studies. Far away from our relatives, we were able to sustain each other and create a great family. Now, we are looking forward to having a baby soon!

PREFACE

This thesis deals with the problem of numerically approximating all of the solutions of a class of nonlinear second order semilinear elliptic boundary value problems with homogeneous boundary conditions on a rectangular domain. It has been noted, see e.g. [5], that there are very few theoretical results concerning how many solutions such a problem may have, or indeed, if there are any solutions at all. This state of affairs is somewhat better in the corresponding case of second order ordinary differential equations, where a number of existence and multiplicity results are available in several papers and books, see, e.g., the references in [1]. The approach taken here is to perform a standard finite difference approximation to the partial derivatives and then to numerically seek all of the solutions to the resulting nonlinear systems of equations. This approach worked very successfully for the ordinary differential equation case when the nonlinearities are of polynomial type, [1]. In this case, the totality of solutions of the polynomial systems of equations was found by regarding the systems in a complex setting and applying a numerical homotopy continuation method. In recent years, a considerable literature and a library of computer programs for finding all complex solutions to polynomial systems has come into being. However, an increasing demand of accuracy for the discretization requires an increasing size of the polynomial system, and this in turn brings about an enormous number of complex solutions, among which are the few real solutions which are actually of interest. To illustrate this point, let us mention the familiar theorem of Bezout [6, 7, 8], which essentially states that the number of (finite) complex solutions, with accounting for multiplicities, can be as much as the product of the degrees of each of the equations. So, for example, ten quadratic equations would generally have $2^{10} = 1024$ complex solutions in \mathbb{C}^{10} . Furthermore, even among the real solutions, there may be spurious solutions which arise as numerical artifacts

and do not converge to solutions of the boundary value problem.

The above drawbacks were treated in [1] by means of two remedial steps. The first step was to discard the obviously irrelevant solutions, including the real solutions which did not exhibit properties which theoretical results showed must hold, for example, symmetry properties. The second remedy was to start with a crude mesh (and hence a low dimensional system) and then to introduce continuously a new mesh point via a mesh deformation. Assuming that the solutions have been obtained for a uniform mesh with, say n points, a new point was introduced, for example, at the right boundary and this point was then allowed to be moved leftward until a uniform mesh with $(n + 1)$ points is achieved. This device was suggested in [9] and was implemented in [1]. It uses numerical continuation in yet another way since the homotopy parameter now is used to deform the mesh size in the difference equations. Starting points for solutions when the new point is introduced are simply the zero points of a single polynomial equation, which are generally easy to find. For example, one may apply an algorithm for finding the eigenvalues of the corresponding companion matrix.

In essence, it would seem straightforward to extend the ideas used in [1] to a corresponding case of partial differential equations in two dimensions. One new issue which needs to be confronted is that of introducing mesh points in a manner which does not require a large number of new points, due to the fact that the number of solutions would grow too rapidly. Such a moving mesh is described in Chapter II. The approach proceeds as follows. Suppose that the solutions have been found for a uniform square mesh in the square domain. In a fashion as described in the previous paragraph, a new point is introduced at the right boundary of the first row of mesh points and is allowed to move leftward until the points in the first row are again equally spaced. The same thing is done for each of the rows and then for each of the columns, this time by introducing a point at the top of each column until once again a uniform square mesh is attained. The entire procedure requires a careful handling of the underlying stiffness matrices for the moving meshes. Now however,

the resulting meshes do not generally exhibit symmetries even at termination of introducing the moving mesh-point unless the mesh is once again symmetric too. So symmetry conditions for discarding spurious solutions need to be correspondingly adapted. On the other hand, since the boundary value problem is being considered on a rectangular domain, whatever symmetric solutions arise are likely to occur on multiple homotopy paths. It turns out that, in a natural way there arises the phenomenon of bifurcation of homotopy paths. For the class of problems being considered here, the homotopy paths are the solutions of a system of equations of the form $H(z, t) = 0$ where $H : \mathbb{C}^n \times [0, 1] \rightarrow \mathbb{C}^n$ is analytic in the z -variables. In Chapter I, we show that in the above analytic setting any turning point of a homotopy path is necessarily also a bifurcation point; we also show that the bifurcating directions are orthogonal. Thus the tangent vectors at the bifurcation point are orthogonal and the numerical continuation at such points is conveniently handled. These results also offer other advantages particularly in handling some of the 'endgame' issues at the termination of the homotopy. Since numerical continuation is arising in several different ways in this thesis, (as homotopy methods for systems, as mesh deformations for difference methods, and as intrinsic bifurcation problems, in Chapter III), a recapitulation of numerical continuation is also presented in Chapter I.

In a paper by Breuer, McKenna and Plum [5], a particular boundary value problem involving a quadratic nonlinearity on a square domain is studied. The authors prove a theorem showing the existence of at least four solutions for a particular value of a parameter arising in the equation. They conjecture a specific bifurcation diagram and suggest that more than four solutions may occur as the parameter is increased. In Chapter III, the methods presented in the thesis are applied to verify the bifurcation diagram and to demonstrate that no more than four solutions occur over a large positive range of the parameter values.

In Chapter IV, a generalization is made which allows nonpolynomial nonlinearities in the partial differential equation. This changes matters in two fundamental ways. First of all, one does not any longer know how many solutions the discretized system has, or indeed,

if there are any, or whether there are only finitely many solutions. The second issue arises when one seeks the solutions of a non-polynomial equation at the introduction of a new mesh point. In the polynomial case, the starting solutions were obtained as the eigenvalues of the corresponding companion matrix. Since there may now be infinitely many complex solutions, it is necessary to seek solutions in a bounded region, which is taken to be a compact rectangular $(2d)$ -cell, where d represents the dimension of the equation. The real solutions are then found by means of a cellular exclusion algorithm. Cellular exclusion algorithms have recently been studied by several authors, see e.g. [10] and the reference therein. With cellular exclusion methods, all real solutions within a rectangular n -cell can be found to systems of equations, provided the nonlinearity satisfies some very general conditions. This approach is applied to the familiar Bratu equation in one dimension. For completeness, the cellular exclusion method is reviewed.

The present study has been restricted to a special, but familiar class of boundary value problems mainly to illustrate the effectiveness of the techniques presented. One may readily envision generalizing the differential operator, the domain and the boundary conditions. Moreover, other discretization methods, such as finite element methods may also be envisioned. Now the matter of introducing a low number of elements presents other issues which would be worthy of investigation. It should be emphasized, that the methods presented here are not meant to provide fast and accurate methods for solving partial differential equations. Rather, the idea is to glean reliable information concerning the number and qualitative properties of solutions. The approximations which are obtained may however, be used as starting values to obtain more accurate solutions on much finer meshes.

TABLE OF CONTENTS

SIGNATURE	ii
ABSTRACT OF DISSERTATION	iii
ACKNOWLEDGEMENTS	iv
PREFACE	vi
LIST OF TABLES	xii
LIST OF FIGURES	xiii
I NUMERICAL CONTINUATION FOR ANALYTIC MAPS	1
1.1 Homotopy Numerical Continuation	2
1.2 Homotopy Continuation for 1-D ordinary differential equations	8
1.3 Arclength Continuation and Turning Points	14
1.4 Possible Tracked Paths and the ‘End Game’ of Tracking	23
II HOMOTOPY CONTINUATION FOR 2-D PARTIAL DIFFERENTIAL EQUATIONS	26
2.1 Difficulties for 1-D ODEs and 2-D PDEs	27
2.2 Details on the 2-D grid for $\Delta u = f(\lambda, x, y, u, u_x, u_y)$	30
2.3 Introducing a point on the l^{th} row. Approximating the derivatives.	32
2.3.1 Approximating the Laplacian at all the interior points except the ones on the rows $l - 1, l$, and $l + 1$	34
2.3.2 Approximating the Laplacian at the interior points on the $(l - 1)^{th}$ row	35
2.3.3 Approximating the Laplacian at the interior points on the l^{th} row, except at the new point (\blacklozenge)	36
2.3.4 Approximating the Laplacian at the new point (\blacklozenge) introduced on the l^{th} row	37
2.3.5 Approximating the Laplacian at the interior points on the $(l + 1)^{th}$ row	38
2.3.6 The discretized equations for $\Delta u = f(\lambda, x, y, u, u_x, u_y)$ necessary to build the homotopy.	39

2.4	The homotopy function for $\Delta u = f(\lambda, x, y, u, u_x, u_y)$	42
2.5	Algorithm for the moving mesh-grid	49
2.6	Refining a solution	50
2.7	Numerical results for polynomial right hand side	52
2.7.1	Numerical results for $\Delta u = -(1 + u^2)$	52
2.7.2	Numerical results for $\Delta u = -\lambda(1 + u^2)$, $\lambda \in \{9, 10\}$	64
2.7.3	The need for complex solutions?	64
III	SOLUTION OF AN OPEN CONJECTURE	66
IV	HOMOTOPY CONTINUATION FOR NONPOLYNOMIAL NONLINEAR- ITY	69
4.1	Bratu Problem and the Necessity of Exclusion Tests	70
4.2	Exclusion Tests	73
4.2.1	Introduction to Exclusion Algorithms	73
4.2.2	Dominant Functions	77
4.2.3	Exclusion Tests Using Dominant Functions	79
4.3	The Solutions for the Bratu Problem	81
V	CONCLUSIONS	85
VI	APPENDICES	88
6.1	Appendix A: Mountain Pass Algorithm	88
6.2	Appendix B: Manual For Our Toolbox	91
6.3	Appendix C: A version of a ‘tracker’ file used to solve the problem (33)	101

LIST OF TABLES

1	The number of solutions for $\Delta u = -(1 + u^2)$ on $\Omega = [0, 1]^2$ with zero Dirichlet boundary conditions. Homotopy with continuation in t was used to track them from $t = 1$ to $t = 0$; if no convergence for a solution, then the gamma-trick was used instead to track it.	56
2	The number of solutions for $\Delta u = -(1 + u^2)$ on $\Omega = [0, 1]^2$ with zero Dirichlet boundary conditions. Arclength continuation was used to track all the solutions of the homotopy associated to the discretization of this problem.	62
3	The number of solutions for $u_{xx} + \lambda \exp(u) = 0$ with zero Dirichlet boundary conditions on $[0, 1]$ for $\lambda = 3$. Homotopy with continuation in t was used to track them from $t = 1$ to $t = 0$; the exclusion algorithm (46) was used to solve (53) in the starting box interval given by $m_\sigma = (10, 0)$ and $r_\sigma = (10, 15)$	82
4	The number of solutions for $u_{xx} + \lambda \exp(u) = 0$ with zero Dirichlet boundary conditions on $[0, 1]$ for $\lambda = 3$. Homotopy with continuation in t was used to track them from $t = 1$ to $t = 0$; the exclusion algorithm (46) was used to solve (53) in the starting box-interval given by $m_\sigma = (5, 0)$ and $r_\sigma = (5, 5)$	83
5	The number of intervals obtained at each bisection level for a time when exclusion algorithm was used to obtain the results from Table 3.	83

LIST OF FIGURES

1	Schematic of the prediction (Euler) and correction (Newton) for path-tracking.	4
2	Possible paths in our tracking.	24
3	Path going to infinity.	25
4	Introducing a new point (\blacklozenge) on the l^{th} row: the picture at the initial step $t = 1$ of the homotopy (<i>left picture</i>), the picture at an intermediate step $0 < t < 1$ of the homotopy (<i>central picture</i>), and the picture at the final step $t = 0$ of the homotopy (<i>right picture</i>).	32
5	Introducing a new point (\blacklozenge) on the l^{th} row: the general case of the homotopy: $0 < t < 1$. Approximating the Laplacian at all the points, except the ones on the rows $l - 1, l$, and $l + 1$	33
6	Introducing a new point (\blacklozenge) on the l^{th} row. Approximating the Laplacian at the points on the $(l - 1)^{th}$ row.	35
7	Introducing a new point (\blacklozenge) on the l^{th} row. Approximating the Laplacian at the points on the l^{th} row <i>except at the new point</i> (\blacklozenge).	36
8	Introducing a new point (\blacklozenge) on the l^{th} row. Approximating the Laplacian <i>at the new point</i> (\blacklozenge).	37
9	Introducing a new point (\blacklozenge) on the l^{th} row. Approximating the Laplacian at the points on the $(l + 1)^{th}$ row.	38
10	Introducing a new point (\blacklozenge) on the l^{th} row: the general case of the homotopy, $0 < t < 1$	39
11	Refining a mesh grid on a box-interval: 2×2 , refined to 5×5 , and then to 11×11	51
12	The bifurcation diagram for $u'' = -\lambda(1 + u^2)$ with zero Dirichlet boundary conditions.	52
13	The two solutions with 150 interior grid-points for $u''(t) = -3(1 + u(t)^2)$ with zero boundary conditions.	53
14	Six solutions for $\Delta u = -(1 + u^2)$ with 4×4 interior points.	57
15	The six solutions for $\Delta u = -(1 + u^2)$ with 4×4 interior points refined to 9×9	58
16	The six solutions for $\Delta u = -(1 + u^2)$ with 4×4 interior points refined to 19×19	58
17	The six solutions for $\Delta u = -(1 + u^2)$ with 4×4 interior points refined to 39×39	59

18	The six solutions for $\Delta u = -(1 + u^2)$ with 4×4 interior points refined to 79×79	59
19	The bifurcation diagram for $\Delta u = -\lambda(1+u^2)$ with zero Dirichlet boundary conditions.	60
20	The four real solutions for $\Delta u + u^2 = 800 \sin \pi x \sin \pi y$	67
21	The bifurcation diagram for $\Delta u + u^2 = \lambda \sin \pi x \sin \pi y$ with Dirichlet boundary conditions.	68
22	The number of real solution for (40).	71
23	The bifurcation diagram for the Bratu Problem.	71
24	Illustration of Bisection Levels for a box-interval $\sigma \subset \mathbb{R}^2$	75
25	The two real solutions for the Bratu problem (49) refined from a mesh with 10 interior points to one with 1407 interior points.	84

CHAPTER I

NUMERICAL CONTINUATION FOR ANALYTIC MAPS

The problem of numerically finding all the solutions for second order ordinary differential equations was successfully approached by the authors in [1] for the case of polynomial nonlinearities. The first section of this chapter will familiarize us with the notion of the homotopy and, since numerical continuation will be arising in several different ways in this thesis, a recapitulation of it is also presented here. A path tracking algorithm (necessary to track all the solutions of such a function) with some useful choices for implementing some of its steps will be presented at the end of this section.

In the next section, we summarize the theory from [1] and present the algorithm used by the authors to numerically find all the solutions for this particular case of problems.

The need for handling turning points for some tracked paths has become very important in our attempt to generalize the method to a class of nonlinear second order semilinear elliptic boundary value problems with homogeneous boundary conditions on a rectangular domain. Therefore, we review the concept of arclength continuation in the third section and prove two important results (generalizations of two theorems from [11]) characterizing turning points of homotopy paths for analytic maps as bifurcation points.

In the last part of this chapter we present all the possible paths we can meet in our tracking, and also some better choices for the ‘End Game’ of tracking.

1.1 Homotopy Numerical Continuation

In this section we will first present an illustrative example which gives the basic ideas of how Homotopy Numerical Continuation can be used to find the roots of a polynomial. Then, we will describe a path tracking algorithm that will help us in tracking the solutions of the homotopy as its t -parameter goes from 1 to 0, as well as some choices for implementing some of its steps.

Suppose we want to find the roots of a polynomial $p(z)$ of degree d of the form

$$p(z) = z^d + a_{d-1}z^{d-1} + \dots + a_0.$$

For this, we first consider $z^d - 1 = 0$ for which we already know the roots

$$z_k^* = e^{2\pi ki/d}, \quad k = 1, \dots, d, \quad \text{where } i = \sqrt{-1}.$$

Now, we form the homotopy

$$H(z, t) := t(z^d - 1) + (1 - t)p(z).$$

At $t = 1$ we have the system $H(z, 1) = z^d - 1$ with known roots, and at $t = 0$, we have the system $H(z, 0) = p(z)$ with the roots we want to find. We seek to numerically track the solution paths from $t = 1$ to $t = 0$.

First, observe that each solution path $z_k^*(t)$ satisfies the Davidenko differential equation (see [11]) for all t :

$$H_z(z_k^*(t), t) \frac{dz_k^*(t)}{dt} + H_t(z_k^*(t), t) = 0 \quad (1)$$

For our particular polynomial case, this implies that

$$\frac{dz_k^*(t)}{dt} = -\frac{H_t(z_k^*(t), t)}{H_z(z_k^*(t), t)} = -\frac{z_k^*(t)^d - 1 - p(z_k^*(t))}{tdz_k^*(t)^{d-1} + (1 - t)p'(z_k^*(t))} \quad (2)$$

Therefore, we can find $z_k^*(t)$ as a solution of an ordinary differential equation with initial value given for $z_k^*(1)$.

Another idea (more numerically stable, and which also takes advantage of the fact that the solution paths satisfy the equation $H(z(t), t) = 0$) is to use a simple path-tracking as described in the next algorithm:

Simple Path-Tracking Algorithm

- Begin
- Set up a grid (uniform for example) t_0, \dots, t_N , $h = \frac{1}{N}$, and $t_j = (N - j)h$
- For each $k = 1, \dots, d$ do
 - (1) initialize $w_0 = z_k(1)$
 - (2) for each $j = 1, \dots, N$ do
 - i. use one step of Euler's method to define $w = w_j - \frac{1}{2w_j}h$
 - ii. find the solution w_{j+1} of $H(z, t_j) = 0$ using Newton-Raphson's methods with start value w .
- End.

A main problem here is that we might have multiple roots or no solutions for the homotopy and the Newton's method does not work so well in these cases. Let's take, for instance, $p(z) = 3 - z^2$ and

$$H(z, t) = t(z^2 - 1) + (1 - t)(3 - z^2).$$

It is clear that we have some trouble at $t = 3/4$ (because $H(z, 3/4) = (1/2)z^2$ has a double root) and at $t = 1/2$ (because $H(z, 1/2) = 1$ has no solution). We can eliminate these problems by using the **gamma-trick**: introduce a random angle $\theta \in [-\pi, \pi]$ and modify the homotopy function from above to

$$H(z, t) = te^{i\theta}(z^d - 1) + (1 - t)p(z) = 0$$

where $i = \sqrt{-1}$. Note that at $t = 1$ and $t = 0$, we have the same starting and ending points respectively, but now, due to the complex factor $\gamma = e^{i\theta}$, the paths are well-behaved for all $t \in [0, 1]$.

The heart of any numerical continuation method is its path-tracking algorithm, for which we can use a predictor/corrector method based on having an explicit homotopy $H(z(t), t)$. Such a method is highly preferred because the corrector step avoids the build-up error which often accumulates in a numerical ODE solver (see [2]).

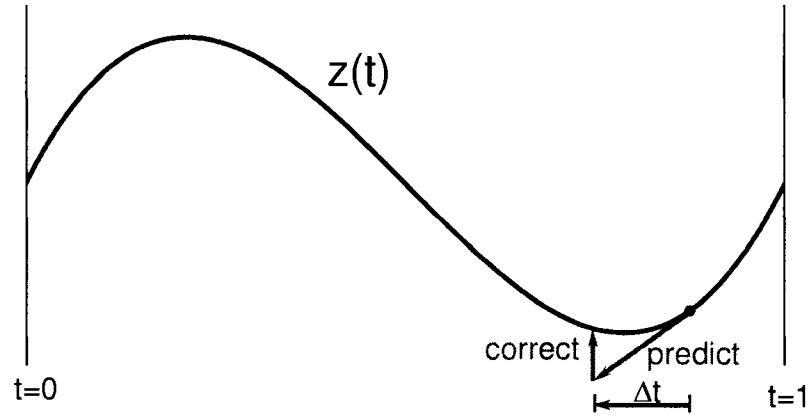


Figure 1: Schematic of the prediction (Euler) and correction (Newton) for path-tracking.

Basic prediction and correction can both be accomplished by considering a local model of the homotopy function via its Taylor series:

$$H(z + \Delta z, t + \Delta t) = H(z, t) + H_z(z, t)\Delta z + H_t(z, t)\Delta t + H.O.T.$$

If we have a point (z_1, t_1) near the path, i.e. $H(z_1, t_1) \approx 0$, then one can predict to a new solution at $t_1 + \Delta t$ by setting $H(z_1 + \Delta z, t_1 + \Delta t) = 0$ and solving the first order terms to get

$$\Delta z = -H_z^{-1}(z_1, t_1)H_t(z_1, t_1)\Delta t \quad (3)$$

On the other hand, when $H(z_1, t_1)$ is not small enough, one may hold t constant by setting

$\Delta t = 0$ and solving the equation to get

$$\Delta z = -H_z^{-1}(z_1, t_1)H(z_1, t_1) \quad (4)$$

These are what we call the *Euler prediction* and *Newton correction*.

In general (see [11]), if $H : \mathbb{R}^{N+1} \rightarrow \mathbb{R}^N$, and u is a point near the path, i.e. $H(u) \approx 0$ then the prediction to a new solution v is made by taking $v := u + ht(H'(u))$, where $t(H'(u)) \in \mathbb{R}^{N+1}$ is the tangent vector induced by $H'(u)$; if $H(v)$ is not small enough, then correct v by repeating

$$w := v - H'(v)^+ H(v)$$

$$v := w$$

until convergence (here, $H'(v)^+$ is the Moore-Penrose inverse of $H'(v)$).

The main concern of a numerical path-tracking algorithm is deciding which of these to do next and how big a step Δt to use in the predictor.

We will present next a generic path-tracking algorithm (see [11]), in which we assume that the path parameter s (arclength, for example) is strictly monotonic and therefore there are no turning points). This is a consequence of the assumption that the Jacobian is non-singular along the path.

Path-Tracking Algorithm

Given: A full-rank system of equations, $F(\mathbf{v}, s) = 0$; initial point \mathbf{v}_0 at $s_0 = 1$ such that $F(\mathbf{v}_0, s_0) \approx 0$; and initial step length h .

Find: A sequence of points (\mathbf{v}_j, s_j) , $j = 1, 2, \dots, n$ along the path such that $F(\mathbf{v}_j, s_j) \approx 0$, $s_{j+1} < s_j$, and terminating with $s_n = 0$. Return a high accuracy estimate for \mathbf{v}_n .

Procedure: For $j = 1, 2, \dots, n$ do:

- **Prediction:** Predict a solution (\mathbf{u}, s') such that $\|(\mathbf{u}, s') - (\mathbf{v}_{j-1}, s_{j-1})\| \approx h$ with $s' < s_{j-1}$.
- **Correction:** Try to find a corrected solution (\mathbf{w}, s'') in a neighborhood of (\mathbf{u}, s') such that $F(\mathbf{w}, s'') \approx 0$.
- **Update:** If the correction step was successful, update $(\mathbf{v}_j, s_j) := (\mathbf{w}, s'')$ and increment j .
- **Adjust:** Adjust the step length h .
- **Terminate Loop:** Terminate the loop when $s_j = 0$ or non-convergence of the path has been detected.

We present below some useful choices for implementing each of these steps:

Prediction: The simplest predictor is $\mathbf{u} = \mathbf{v}_{j-1}$, but it is better to use a linear or a higher-order prediction. One can use for example, one of the following linear predictors:

- **Secant Predictor** Use the last two points on the path to linearly extrapolate to the next:

$$(\mathbf{u}, s') = (\mathbf{v}_{j-1}, s) + h \frac{\Delta_j}{\|\Delta_j\|}, \text{ where } \Delta_j = (\mathbf{v}_{j-1} - \mathbf{v}_{j-2}, s_{j-1} - s_{j-2})$$

- **Tangent Predictor** Step along the tangent direction:

$$\mathbf{u} = \mathbf{v}_j - \alpha \left(\frac{\partial F}{\partial \mathbf{v}} \right)^{-1} \left(\frac{\partial F}{\partial s} \right)$$

where α is calculated to give the desired step length (this is what we call Euler's method).

Step Length: The step length can be measured by any preferred norm of $(\mathbf{u} - \mathbf{v}_j, s' - s_{j-1})$.

A simple choice is $\|(\mathbf{u} - \mathbf{v}_j, s' - s_{j-1})\| := |s' - s_{j-1}|$.

Correction: One can use the following common corrector: hold s constant, i.e. $s'' = s'$, and compute \mathbf{w} by Newton's method, allowing a fixed number of iterations.

Step Length Adjustment: In case of a failure of the corrector, one can cut the step length by half. In the case of m successive corrections at the current step size, one can double it (a choice of m in between 2 to 5 works well).

Final Step: Near the end of the path-tracking interval, one needs to adjust the step length to land exactly on $s = 0$.

Terminate: Eventually, we must arrive at $s = 0$ or else $|s' - s_{j-1}|$ must become progressively smaller. One should set a minimum threshold for progress in s , below which we declare the path is either diverging or approaching a singularity. One can also terminate if the magnitude of the solutions grows too large.

Refine: Newton's method will work fine for nonsingular endpoints.

1.2 Homotopy Continuation for 1-D ordinary differential equations

Consider the second order boundary value problem on the interval $[a, b] \subset \mathbb{R}$

$$u'' = f(x, u, u') \quad (5)$$

with the boundary conditions $u(a) = \alpha$ and $u(b) = \beta$. Using a central difference approximation with a uniform mesh for example, we can approximate a solution $u(x)$ of (5) by an N -tuple of numbers $(u_1, u_2, \dots, u_N)^T$ such that $u_i \approx u(x_i)$, $\forall i = 1, \dots, N$, where we set $h := \frac{b-a}{N+1}$, $x_i := a + ih$, $\forall i = 0, \dots, N+1$, $u_0 = \alpha$, and $u_{N+1} = \beta$. The discretization of (5) takes the form of the following system \mathcal{D}_N :

$$\mathcal{D}_N \begin{cases} u_0 & - & 2u_1 & + & u_2 & = & h^2 f\left(x_1, u_1, \frac{u_2 - u_0}{2h}\right) \\ \vdots & & \vdots & & \vdots & = & \vdots \\ u_{N-1} & - & 2u_N & + & u_{N+1} & = & h^2 f\left(x_N, u_N, \frac{u_{N+1} - u_{N-1}}{2h}\right) \end{cases}$$

We are seeking all the real solutions of (5) and we know that depending upon the right hand side f , equation (5) may have no solution, a unique solution, multiple solutions, or even infinitely many solutions. There are many theorems that state the existence solutions for such equations, but even when the existence is known, the number of solutions is often not. In [1], the authors studied a relatively secure numerical technique for finding all the solutions for such an equation in the case that $f(x, u, u')$ from (5) is just a polynomial depending on u . The idea is to complexify the problem and find all the solutions (real and complex) of an associated polynomial system $P(z)$ using a homotopy function which will track some known starting solutions to all the solutions of $P(z)$. It is important to remark Bezout's Theorem for Polynomial Systems.

Theorem 1.2.1 (Bezout). *Let $P : \mathbb{C}^n \rightarrow \mathbb{C}^n$, $P(\vec{z}) = \{p_j(z_1, \dots, z_n)\}_{j=1, \dots, n}$ be a system of polynomials with $d_j = \deg p_j$ for any $j = 1, \dots, n$. Then, counting their multiplicity, the number of regular solutions (real and complex) of P is*

$$d = \prod_{j=1}^n d_j$$

Another important result about the number of solutions of a polynomial system can be found in [7]. Also, two extensive surveys of homotopy methods for polynomial systems are given by Li in [12] and [13].

In this section, we will present briefly the idea and the algorithm for finding the solutions of this 1-D problem. The process of finding these solutions can be sketched in four steps:

1. Find all the solutions of the discretization \mathcal{D}_N for some small N .
2. Discard all unreasonable solutions and denote by Ω_N the set of the solutions which are kept.
3. If the mesh size is not sufficiently small or the cardinality of Ω_N has not yet stabilized, then add a mesh point to obtain the discretization \mathcal{D}_{N+1} . Use the solutions in Ω_N to generate solutions of \mathcal{D}_{N+1} and then return to Step 2 to generate Ω_{N+1} .
4. Once the mesh size is sufficiently small and the cardinality of Ω_N becomes stable, refine the solutions to a more consistent grid with a fast nonlinear solver.

As one can imagine, the key step in this process is number 3. To solve this, we first consider the following homotopy function which gives a mesh refinement in continuous deformation:

$$H_{N+1}(u_1, u_2, \dots, u_{N+1}, t) :=$$

$$\begin{bmatrix} u_0 - 2u_1 + u_2 & - & h(t)^2 f\left(x_1(t), u_1, \frac{u_2 - u_0}{2h(t)}\right) \\ & \vdots & \\ u_{N-2} - 2u_{N-1} + u_N & - & h(t)^2 f\left(x_{N-1}(t), u_{N-1}, \frac{u_N - u_{N-2}}{2h(t)}\right) \\ u_{N-1} - 2u_N + U_{N+1}(t) & - & h(t)^2 f\left(x_N(t), u_N, \frac{U_{N+1}(t) - u_{N-1}}{2h(t)}\right) \\ u_N - 2u_{N+1} + U_{N+2}(t) & - & h(t)^2 f\left(x_{N+1}(t), u_{N+1}, \frac{U_{N+2}(t) - u_N}{2h(t)}\right) \end{bmatrix} \quad (6)$$

with

$$\begin{cases} x_i(t) := a + ih(t), & \forall i = 1, \dots, N+1 \\ u_0 := \alpha \\ h(t) := t \left(\frac{b-a}{N+1} \right) + (1-t) \left(\frac{b-a}{N+2} \right) \\ U_{N+1}(t) := (1-t)u_{N+1} + \beta t \\ U_{N+2}(t) := \beta(1-t) \end{cases} \quad (7)$$

Remarks 1.2.2.

- At $t = 0$, H_{N+1} represents the system \mathcal{D}_{N+1} .
- At $t = 1$, H_{N+1} can be interpreted as the system \mathcal{D}_N with a new mesh point having the value u_{N+1} at $x_{N+1} = b$ and a new right-hand boundary having the value $U_{N+2}(1) = 0$ at $x_{N+2} = b + h(1)$.
- There is an incompatibility between the old boundary condition at $x = b$ and the new one at $x = b + h(1)$, but this is accommodated by the presence of both u_{N+1} and U_{N+1} , which are not necessarily equal. As t goes from 1 to 0, the mesh points are squeezed back inside of $[a, b]$ and right hand boundary condition $u(b) = \beta$ is transferred from U_{N+1} to U_{N+2} as U_{N+1} is enforced to equal u_{N+1} , i.e.

$$U_{N+2}(0) = \beta \text{ and } U_{N+1}(0) = u_{N+1}.$$

To find all the solutions of \mathcal{D}_{N+1} , we will use continuation to track the zeros of H_{N+1} as t goes from 1 to 0. At $t = 1$, we have a list Ω_N of solutions $(u_1, u_2, \dots, u_N)^T$ satisfying the first N equations of H_{N+1} , while the final equation is

$$u_N - 2u_{N+1} = h(1)^2 f\left(b, u_{N+1}, \frac{-u_N}{2h(1)}\right), \quad (8)$$

which is the only place where u_{N+1} appears. For each solution $(u_1, u_2, \dots, u_N)^T$ in Ω_N , we use this equation to find the corresponding values of u_{N+1} . These are the starting points of continuation paths leading to solutions of \mathcal{D}_{N+1} . Further information about homotopy numerical continuation will be given in the next chapter.

Remark 1.2.3. This framework will not change in any of its essentials if we prescribe in (7) a different function for U_{N+2} (for example, the constant function $U_{N+2} := \beta$). *The essential feature of U_{N+2} is that it goes to β as t goes from 1 to 0.*

Remark 1.2.4. By the Implicit Function Theorem (IFT), we know that a nonsingular solution $u = u^*$ to $H_{N+1}(u, 1) = 0$ will continue uniquely in an neighborhood of $t = 1$ to a nonsingular solution path $u(t)$ satisfying $H_{N+1}(u(t), t) = 0$ with $u(1) = u^*$. However, this does not guarantee that the path will remain nonsingular all the way to $t = 0$, which is what we require to follow the path reliably with numerical continuation. This difficulty can be passed over using the **gamma-trick** (see [2], Chapter 7).

In our case, it is sufficient to introduce a random $\gamma \in \mathbb{C}$ into the homotopy to obtain the variant:

$$H_{N+1}(u_1, u_2, \dots, u_{N+1}, t) := \begin{bmatrix} \Gamma(t)(u_0 - 2u_1 + u_2) - h(t)^2 f\left(x_1(t), u_1, \frac{u_2 - u_0}{2h(t)}\right) \\ \vdots \\ \Gamma(t)(u_{N-2} - 2u_{N-1} + u_N) - h(t)^2 f\left(x_{N-1}(t), u_{N-1}, \frac{u_N - u_{N-2}}{2h(t)}\right) \\ \Gamma(t)(u_{N-1} - 2u_N) + U_{N+1}(t) - h(t)^2 f\left(x_N(t), u_N, \frac{U_{N+1}(t) - u_{N-1}}{2h(t)}\right) \\ \Gamma(t)(u_N - 2u_{N+1} + \beta) - h(t)^2 f\left(x_{N+1}(t), u_{N+1}, \frac{\beta - u_N}{2h(t)}\right) \end{bmatrix}$$

with

$$\left\{ \begin{array}{l} \Gamma(t) := \gamma^2 t + (1 - t) \\ h(t) := \gamma t \left(\frac{b-a}{N+1} \right) + (1 - t) \left(\frac{b-a}{N+2} \right) \\ U_{N+1}(t) := (1 - t)u_{N+1} + \gamma^2 \beta t \\ x_i(t) := a + ih(t), \quad \forall i = 1, \dots, N + 1 \end{array} \right.$$

Remark 1.2.5. In essence, the gamma-trick rests upon a parametrized Sard's theorem (see e.g. [11]).

Remark 1.2.6. In the case of polynomial nonlinearity when $f(x, u, u')$ from (5) is a real polynomial $p(u)$, we can conveniently obtain the starting points for

$H_{N+1}(u_1, u_2, \dots, u_{N+1}, 1) = 0$ by solving the polynomial equation:

$$u_N - 2u_{N+1} + \beta - \left(\frac{b-a}{N+1} \right)^2 p(u_{N+1}) = 0 \quad (9)$$

for u_{N+1} given u_N from the solutions in Ω_N . All the solutions (real and complex) of (9) can be found using standard available software (for small degree polynomials, the companion matrix may be used to find these zeros).

If we denote $d = \deg p(u)$, then one can see that over the complex numbers we obtain d values of u_{N+1} for every solution in Ω_N . If we suppose that at each stage of the algorithm these will continue to finite, nonsingular solutions of \mathcal{D}_{N+1} , then Ω_N will have d^N entries. Therefore, the number of the solutions for \mathcal{D}_{N+1} grows exponentially as N increases, and hence the need of some filters becomes very important. It is important to remark here that all the real and complex solutions of (9) (and in general of any polynomial equation) can be found using standard available software, e.g. the *roots* command in Matlab which simply involves computing the eigenvalues of the associated companion matrix.

Now, we can write the final version of the algorithm:

Algorithm:

1. For $N = 1$, find Ω_1 (since $H_1(u_1, 1)$ is a polynomial in u_1 and it may be solved by any one-variable method).
2. For $N = 2, 3, \dots$ do the following until some desired behavior occurs:
 - Build $H_N(u_1, u_2, \dots, u_N, t)$.
 - For each solution in Ω_{N-1} , solve the last polynomial equation of $H_N(u_1, u_2, \dots, u_N, t)$ (which is (9)) for u_N , and then form the set S_N of the start solutions for this system $H_N(u_1, u_2, \dots, u_N, t)$.
 - Use numerical continuation to track all the paths beginning at the points in S_N at $t = 1$. The set of all endpoints of these paths will form Ω_N after applying some filters (if desired).
3. If desired, refine the solutions to more consistent grids using a fast nonlinear solver.

Some numerical results can be found in [1].

Remark 1.2.7. The homotopy parameter t brings an extra point into the finite difference discretization, as one can see in the figure from the next page.

In this thesis, we will extend the concept to 2-D partial differential equations and to non-polynomial right hand sides.

1.3 Arclength Continuation and Turning Points

Using just numerical continuation in t to solve $H(t, u) = 0$ is a good idea if turning points do not occur as we track the solutions of H from $t = 1$ to $t = 0$. It was suggested in [1] to use the gamma-trick if such turning points are present. The use of this trick as explained in section 1.2 makes the paths well behaved (no turning points), but in our experiments we observed that in most of the cases that we used it we ended up finding solutions at $t = 0$ that were also found using continuation in t with other different starting solutions at $t = 1$ (we refer to this process as *losing solutions*) (see section 2.7.1 for an example). Therefore, we needed to introduce arclength continuation [11, 14] in our toolbox to track the solutions as t goes from 1 to 0.

Definition 1.3.1. Let $H : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ be sufficiently smooth. Suppose that $c : J \rightarrow \mathbb{R} \times \mathbb{R}^n$ is a smooth curve, defined on an open interval J containing zero, and parametrized (for reasons of simplicity) with respect to arclength such that $H(c(s)) = 0$ for $s \in J$. The point $c(0)$ is called a **bifurcation point** of the equation $H = 0$ if there is an $\epsilon > 0$ such that every neighborhood of $c(0)$ contains zero-points z of H which are not on $c((-\epsilon, \epsilon))$.

Remark 1.3.2. An immediate consequence of this definition is that a bifurcation point $c(0)$ of $H = 0$ must be a singular point of H . Hence, the Jacobian $H'(c(0))$ must have a kernel of dimension at least two.

Definition 1.3.3. Let $H : \mathbb{R}^{n+1} \rightarrow \mathbb{R}^n$ be a sufficiently smooth. A point $\bar{u} \in \mathbb{R}^{n+1}$ is called a **simple bifurcation point** of the equation $H = 0$ if the following conditions hold:

- (1) $H(\bar{u}) = 0$;
- (2) $\dim \ker H'(\bar{u}) = 2$;
- (3) $e^* H''(\bar{u}) \Big|_{(\ker H'(\bar{u}))^2}$ has one positive and one negative eigenvalue, where e spans $\ker H'(\bar{u})^*$.

The following three results are summarized from [11] and furnish a criterion for detecting a simple bifurcation point when traversing a curve c .

Theorem 1.3.4. *Let $H : \mathbb{R}^{n+1} \rightarrow \mathbb{R}^n$ be a sufficiently smooth and $\bar{u} \in \mathbb{R}^{n+1}$ a simple bifurcation point of the equation $H = 0$. Then there exist two smooth curves $c_1(s), c_2(s) \in \mathbb{R}^{n+1}$, parametrized with respect to the arclength s , defined for $s \in (-\epsilon, \epsilon)$ and ϵ sufficiently small, such that the following holds*

- (1) $H(c_i(s)) = 0, i \in \{1, 2\}, s \in (-\epsilon, \epsilon)$;
- (2) $c_i(0) = \bar{u}, i \in \{1, 2\}$;
- (3) $\dot{c}_1(0), \dot{c}_2(0)$ are linearly independent;
- (4) $H^{-1}(0)$ coincides locally with $\text{range}(c_1) \cup \text{range}(c_2)$; more precisely: \bar{u} is not in the closure of $H^{-1}(0) \setminus (\text{range}(c_1) \cup \text{range}(c_2))$.

Lemma 1.3.5. *Let $\bar{u} \in \mathbb{R}^{n+1}$ be a simple bifurcation point of the equation $H = 0$. Under the notations of 1.3.3 and 1.3.4, we obtain*

- (1) $\ker H'(\bar{u}) = \text{span} \{\dot{c}_1(0), \dot{c}_2(0)\}$;
- (2) $e^* H''(\bar{u}) [\dot{c}_i(0), \dot{c}_i(0)] = 0$, for $i \in \{1, 2\}$.

Theorem 1.3.6. *Let $\bar{u} \in \mathbb{R}^{n+1}$ be a simple bifurcation point of the equation $H = 0$. Under the notations of 1.3.3 and 1.3.4, the determinant of the following augmented Jacobian*

$$\det \begin{pmatrix} H'(c_i(s)) \\ \dot{c}_i(s)^* \end{pmatrix}$$

changes sign at $s = 0$ for $i \in \{1, 2\}$.

Now, let $H : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^n$, be a smooth homotopy. The important assumption which we need to make is that $H(t, u)$ is real analytic in the variables u . Hence it is meaningful to replace u in $H(t, u)$ by $w \in \mathbb{C}^n$. In the following we use the notation $w = u + iv$ for

$w \in \mathbb{C}^n$, where $u, v \in \mathbb{R}^n$ denote the real and the imaginary parts of w respectively. Note that $\overline{H(t, \bar{w})} = H(t, w)$ since H is real analytic. Let us define now the real and imaginary parts $H^r, H^i : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n \times \mathbb{R}^n$ by

$$\begin{aligned} H^r(t, u, v) &:= \frac{1}{2} (H(t, w) + H(t, \bar{w})), \\ H^i(t, u, v) &:= \frac{-i}{2} (H(t, w) - H(t, \bar{w})), \end{aligned} \quad (10)$$

and the map $\hat{H} : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n \times \mathbb{R}^n$ by

$$\hat{H}(t, u, v) := \begin{pmatrix} H^r(t, u, v) \\ -H^i(t, u, v) \end{pmatrix} \quad (11)$$

The numerical aspect then consists of tracing a smooth curve $\hat{c} : s \mapsto (t(s), u(s), v(s))$ in $\hat{H}^{-1}(0)$, where for simplicity s is an arclength parameter. Differentiating $\hat{H}(t(s), u(s), v(s))$ with respect to s yields

$$\begin{pmatrix} \hat{H}_t & \hat{H}_u & \hat{H}_v \end{pmatrix} \begin{pmatrix} \dot{t} \\ \dot{u} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad (12)$$

From (10) we obtain the Cauchy-Riemann equations

$$H_v^r = -H_u^i \quad \text{and} \quad H_v^i = H_u^r. \quad (13)$$

and therefore

$$\hat{H}_{(u,v)} = \begin{pmatrix} H_u^r & -H_u^i \\ -H_u^i & -H_u^r \end{pmatrix}$$

is symmetric. Furthermore, if μ is an eigenvalue of $\hat{H}_{(u,v)}$ with the corresponding eigenvector $\begin{pmatrix} \tilde{u} \\ \tilde{v} \end{pmatrix}$, then so is $-\mu$ with a corresponding eigenvector $\begin{pmatrix} \tilde{v} \\ -\tilde{u} \end{pmatrix}$. Hence, the eigenvalues of $\hat{H}_{(u,v)}$ occur in symmetric pairs about zero, and $\det \hat{H}_{(u,v)}$ never changes sign.

By using the Cauchy-Riemann equations (13) and augmenting (12) in an obvious way we obtain

$$\begin{pmatrix} \dot{t} & \dot{u}^* & \dot{v}^* \\ \hat{H}_t & \hat{H}_u & \hat{H}_v \end{pmatrix} \begin{pmatrix} \dot{t} & 0^* & 0^* \\ \dot{u} & \text{Id} & 0 \\ \dot{v} & 0 & \text{Id} \end{pmatrix} = \begin{pmatrix} 1 & \dot{u}^* & \dot{v}^* \\ 0 & H_u^r & -H_u^i \\ 0 & -H_u^i & -H_u^r \end{pmatrix} \quad (14)$$

and therefore

$$t \det \begin{pmatrix} \dot{t} & \dot{u}^* & \dot{v}^* \\ \hat{H}_t & \hat{H}_u & \hat{H}_v \end{pmatrix} = \det \begin{pmatrix} H_u^r & -H_u^i \\ -H_u^i & -H_u^r \end{pmatrix} = \det \hat{H}_{(u,v)}. \quad (15)$$

Consequently, if U is a neighborhood of a parameter value \bar{s} such that $\hat{c}(s)$ are regular points of \hat{H} for $s \in U \setminus \{\bar{s}\}$, then (15) shows that $\dot{t}(\hat{c}(s))$ changes sign at $s = \bar{s}$ if and only if

$$\det \begin{pmatrix} \dot{t} & \dot{u}^* & \dot{v}^* \\ \hat{H}_t & \hat{H}_u & \hat{H}_v \end{pmatrix} \bigg|_{\hat{c}(s)} \quad (16)$$

does. Hence, a turning point of \hat{c} with respect to the t parameter is also a bifurcation point of the equation $\hat{H} = 0$.

Theorem 1.3.7. *Let $\hat{c}(s) = (t(s), u(s), v(s))$ be a solution curve of $\hat{H}^{-1}(0)$. Suppose $\hat{c}(\bar{s})$ is a simple turning point of the equation $\hat{H} = 0$, i.e. $\dot{t}(\bar{s}) = 0$, $\ddot{t}(\bar{s}) \neq 0$, and the augmented Jacobian from (16) has minimum rank deficiency. Then $\hat{c}(\bar{s})$ is a simple bifurcation point of the equation $\hat{H} = 0$.*

Proof. Using the Cauchy Riemann equations (13), the $(2n + 1) \times (2n + 1)$ augmented Jacobian for \hat{H} takes the form

$$\begin{pmatrix} \dot{t} & \dot{u}^* & \dot{v}^* \\ \hat{H}_t & \hat{H}_u & \hat{H}_v \end{pmatrix} \bigg|_{\hat{c}(\bar{s})} = \begin{pmatrix} 0 & \dot{u}^* & \dot{v}^* \\ H_t^r & H_u^r & -H_u^i \\ -H_t^i & -H_u^i & -H_u^r \end{pmatrix} \bigg|_{\hat{c}(\bar{s})}. \quad (17)$$

Differentiating $\hat{H}(\hat{c}(s)) = 0$ gives:

$$\begin{pmatrix} \hat{H}_t(\hat{c}(s)) & \hat{H}_u(\hat{c}(s)) & \hat{H}_v(\hat{c}(s)) \end{pmatrix} \begin{pmatrix} \dot{t}(s) \\ \dot{u}(s) \\ \dot{v}(s) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

and therefore, at $s = \bar{s}$ we obtain:

$$\begin{pmatrix} H_u^r(\hat{c}(\bar{s})) & -H_u^i(\hat{c}(\bar{s})) \\ -H_u^i(\hat{c}(\bar{s})) & -H_u^r(\hat{c}(\bar{s})) \end{pmatrix} \begin{pmatrix} \dot{u}(\bar{s}) \\ \dot{v}(\bar{s}) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

One can easily see now that also the following holds:

$$\begin{pmatrix} H_u^r(\hat{c}(\bar{s})) & -H_u^i(\hat{c}(\bar{s})) \\ -H_u^i(\hat{c}(\bar{s})) & -H_u^r(\hat{c}(\bar{s})) \end{pmatrix} \begin{pmatrix} \dot{v}(\bar{s}) \\ -\dot{u}(\bar{s}) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

and therefore $(0, \dot{u}(\bar{s}), \dot{v}(\bar{s}))^T$ and $(0, \dot{v}(\bar{s}), -\dot{u}(\bar{s}))^T$ are two linearly independent vectors from $\ker \hat{H}'(\hat{c}(\bar{s}))$ (it is easy to check the linear independence). Since we know that the augmented Jacobian has minimum rank deficiency, we can conclude that $\ker \hat{H}'(\hat{c}(\bar{s}))$ is spanned by these two vectors and therefore the first two conditions from 1.3.3 are satisfied.

It is also important to remark that the rank of the augmented Jacobian from (17) is $2n$, since $(0, \dot{v}(\bar{s}), -\dot{u}(\bar{s}))^T$ spans its kernel.

It remains now to show the non-degeneracy condition for $\hat{H}''(\hat{c}(\bar{s}))$ (the third condition from 1.3.3). Let's first denote $\tilde{c} = \hat{c}(\bar{s})$. Let $(e_1, e_2)^T$ be in the $\ker \hat{H}'(\tilde{c})^*$, i.e.

$$\hat{H}'(\tilde{c})^* \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \iff \begin{pmatrix} H_t^r(\tilde{c})^* & -H_t^i(\tilde{c})^* \\ H_u^r(\tilde{c})^* & -H_u^i(\tilde{c})^* \\ -H_u^i(\tilde{c})^* & -H_u^r(\tilde{c})^* \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

This can be rewritten as:

$$\begin{cases} H_t^r(\tilde{c})^* e_1 - H_t^i(\tilde{c})^* e_2 = 0 \\ H_u^r(\tilde{c})^* e_1 - H_u^i(\tilde{c})^* e_2 = 0 \\ H_u^i(\tilde{c})^* e_1 + H_u^r(\tilde{c})^* e_2 = 0 \end{cases} \quad (18)$$

Consider now the following bilinear form:

$$(\xi, \eta) \longmapsto B(\xi, \eta) \quad (19)$$

where

$$B(\xi, \eta) = (e_1^*, e_2^*) \hat{H}''(\tilde{c}) \left[\xi \begin{pmatrix} 0 \\ \dot{u}(\bar{s}) \\ \dot{v}(\bar{s}) \end{pmatrix}, \eta \begin{pmatrix} 0 \\ \dot{v}(\bar{s}) \\ -\dot{u}(\bar{s}) \end{pmatrix} \right]$$

Knowing that $\dot{t}(s) = 0$, the bilinear form can now be rewritten as:

$$\begin{aligned}
B(\xi, \eta) &= (e_1^*, e_2^*) \left(\hat{H}_t(\tilde{c})\ddot{t}(\bar{s}) + \hat{H}_u(\tilde{c})\ddot{u}(\bar{s}) + \hat{H}_v(\tilde{c})\ddot{v}(\bar{s}) \right) + \\
&\quad + (e_1^*, e_2^*) \left(\hat{H}_{uu}(\tilde{c}) [\xi\dot{u}(\bar{s}), \eta\dot{v}(\bar{s})] + \hat{H}_{vv}(\tilde{c}) [\xi\dot{v}(\bar{s}), -\eta\dot{u}(\bar{s})] \right) + \\
&\quad + (e_1^*, e_2^*) \hat{H}_{uv}(\tilde{c}) ([\xi\dot{u}(\bar{s}), -\eta\dot{u}(\bar{s})] + [\xi\dot{v}(\bar{s}), \eta\dot{v}(\bar{s})]) \\
\\
B(\xi, \eta) &= (e_1^* H_t^r(\tilde{c}) - e_2^* H_t^i(\tilde{c})) \ddot{t}(\bar{s}) + \\
&\quad + (e_1^* H_u^r(\tilde{c}) - e_2^* H_u^i(\tilde{c})) \ddot{u}(\bar{s}) + \\
&\quad + (e_1^* H_v^r(\tilde{c}) - e_2^* H_v^i(\tilde{c})) \ddot{v}(\bar{s}) + \\
&\quad + \xi\eta \left\{ (e_1^* H_{uu}^r(\tilde{c}) - e_2^* H_{uu}^i(\tilde{c})) - (e_1^* H_{vv}^r(\tilde{c}) - e_2^* H_{vv}^i(\tilde{c})) \right\} [\dot{u}(\bar{s}), \dot{v}(\bar{s})] + \\
&\quad + \xi\eta (e_1^* H_{uv}^r(\tilde{c}) - e_2^* H_{uv}^i(\tilde{c})) \{ [\dot{v}(\bar{s}), \dot{v}(\bar{s})] - [\dot{u}(\bar{s}), \dot{u}(\bar{s})] \}
\end{aligned}$$

Differentiating the Cauchy Riemann equations (13) gives:

$$H_{vv}^r = -H_{uu}^r, \quad H_{vv}^i = -H_{uu}^i, \quad H_{uv}^r = -H_{uv}^i, \quad H_{uv}^i = H_{uv}^r. \quad (20)$$

Using (13), (18), and (20), we obtain:

$$B(\xi, \eta) = \xi\eta K$$

where

$$\begin{aligned}
K &= 2 (e_1^* H_{uu}^r(\tilde{c}) - e_2^* H_{uu}^i(\tilde{c})) [\dot{u}(\bar{s}), \dot{v}(\bar{s})] + \\
&\quad + (e_1^* H_{uv}^i(\tilde{c}) + e_2^* H_{uv}^r(\tilde{c})) \{ [\dot{u}(\bar{s}), \dot{u}(\bar{s})] - [\dot{v}(\bar{s}), \dot{v}(\bar{s})] \}
\end{aligned}$$

It is clear that the bilinear form $B(\xi, \eta)$ from (19) has one positive and one negative eigenvalue if and only if the constant K is non-zero. To show this, let's differentiate twice the equation $(e_2^*, -e_1^*) \hat{H}(\hat{c}(s)) = 0$:

$$\begin{aligned}
0 &= (e_2^*, -e_1^*) \left(\hat{H}_t(\tilde{c})\ddot{t}(\bar{s}) + \hat{H}_u(\tilde{c})\ddot{u}(\bar{s}) + \hat{H}_v(\tilde{c})\ddot{v}(\bar{s}) \right) + \\
&\quad + (e_2^*, -e_1^*) \left(\hat{H}_{uu}(\tilde{c}) [\dot{u}(\bar{s}), \dot{u}(\bar{s})] + \hat{H}_{vv}(\tilde{c}) [\dot{v}(\bar{s}), \dot{v}(\bar{s})] \right) + \\
&\quad + 2 (e_2^*, -e_1^*) \hat{H}_{uv}(\tilde{c}) [\dot{u}(\bar{s}), \dot{v}(\bar{s})]
\end{aligned}$$

Doing the calculations, this becomes:

$$\begin{aligned}
0 = & (e_2^* H_t^r(\tilde{c}) + e_1^* H_t^i(\tilde{c})) \ddot{t}(\bar{s}) + \\
& + (e_2^* H_u^r(\tilde{c}) + e_1^* H_u^i(\tilde{c})) \ddot{u}(\bar{s}) + (e_2^* H_v^r(\tilde{c}) + e_1^* H_v^i(\tilde{c})) \ddot{v}(\bar{s}) + \\
& + (e_2^* H_{uu}^r(\tilde{c}) + e_1^* H_{uu}^i(\tilde{c})) [\dot{u}(\bar{s}), \dot{u}(\bar{s})] + (e_2^* H_{vv}^r(\tilde{c}) + e_1^* H_{vv}^i(\tilde{c})) [\dot{v}(\bar{s}), \dot{v}(\bar{s})] + \\
& + 2 (e_2^* H_{uv}^r(\tilde{c}) + e_1^* H_{uv}^i(\tilde{c})) [\dot{u}(\bar{s}), \dot{v}(\bar{s})]
\end{aligned}$$

Taking into account (13), (18), and (20), the previous identity simplifies to:

$$\begin{aligned}
0 = & (e_2^* H_t^r(\tilde{c}) + e_1^* H_t^i(\tilde{c})) \ddot{t}(\bar{s}) + (e_2^* H_{uu}^r(\tilde{c}) + e_1^* H_{uu}^i(\tilde{c})) [\dot{u}(\bar{s}), \dot{u}(\bar{s})] - \\
& - (e_2^* H_{uu}^r(\tilde{c}) + e_1^* H_{uu}^i(\tilde{c})) [\dot{v}(\bar{s}), \dot{v}(\bar{s})] + 2 (-e_2^* H_{uu}^i(\tilde{c}) + e_1^* H_{uu}^r(\tilde{c})) [\dot{u}(\bar{s}), \dot{v}(\bar{s})]
\end{aligned}$$

which can be rewritten as

$$K + (e_2^* H_t^r(\tilde{c}) + e_1^* H_t^i(\tilde{c})) \ddot{t}(\bar{s}) = 0$$

Since $\ddot{t}(\bar{s}) \neq 0$, to conclude that $K \neq 0$, we need to prove that $e_2^* H_t^r(\tilde{c}) + e_1^* H_t^i(\tilde{c}) \neq 0$. Suppose by contradiction that $e_2^* H_t^r(\tilde{c}) + e_1^* H_t^i(\tilde{c}) = 0$. Then, since $(e_1, e_2)^T \in \ker \hat{H}(\hat{c}(\bar{s}))^*$, we can easily see that

$$\begin{pmatrix} H_t^r(\tilde{c})^* & -H_t^i(\tilde{c})^* \\ H_u^r(\tilde{c})^* & -H_u^i(\tilde{c})^* \\ -H_u^i(\tilde{c})^* & -H_u^r(\tilde{c})^* \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} H_t^r(\tilde{c})^* & -H_t^i(\tilde{c})^* \\ H_u^r(\tilde{c})^* & -H_u^i(\tilde{c})^* \\ -H_u^i(\tilde{c})^* & -H_u^r(\tilde{c})^* \end{pmatrix} \begin{pmatrix} e_2 \\ -e_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

and hence

$$\begin{pmatrix} 0 & H_t^r(\tilde{c})^* & -H_t^i(\tilde{c})^* \\ \dot{u}(\bar{s}) & H_u^r(\tilde{c})^* & -H_u^i(\tilde{c})^* \\ \dot{v}(\bar{s}) & -H_u^i(\tilde{c})^* & -H_u^r(\tilde{c})^* \end{pmatrix} \begin{pmatrix} 0 \\ e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 & H_t^r(\tilde{c})^* & -H_t^i(\tilde{c})^* \\ \dot{u}(\bar{s}) & H_u^r(\tilde{c})^* & -H_u^i(\tilde{c})^* \\ \dot{v}(\bar{s}) & -H_u^i(\tilde{c})^* & -H_u^r(\tilde{c})^* \end{pmatrix} \begin{pmatrix} 0 \\ e_2 \\ -e_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

which implies that the augmented Jacobian from (17) has rank deficiency at least two; this

is a contradiction since we already remarked before that its rank is exactly $2n$. \square

Let us show now that at such a bifurcation point, the two solution branches have the same curvatures (in absolute value), but of opposite signs, and therefore, a choice of following branches is available so that the t co-ordinate is decreasing.

Theorem 1.3.8. *Under the assumptions of 1.3.7, let us now denote the two bifurcating solution curves of $\hat{H}^{-1}(0)$ by $\hat{c}_i(s) := (t_i(s), u_i(s), v_i(s))$, $i \in \{1, 2\}$. The curves are defined for s near \bar{s} and $\tilde{c} := \hat{c}_1(\bar{s}) = \hat{c}_2(\bar{s})$ is the bifurcation point. Then $\ddot{t}_1(\bar{s}) = -\ddot{t}_2(\bar{s})$.*

Proof. Let us denote $\hat{c}(s) := (t(s), u(s), v(s))$ for either of the two solution curves \hat{c}_1 or \hat{c}_2 . Differentiating $\hat{H}(\hat{c}(s)) = 0$ twice with respect to s and taking $\dot{t}(\bar{s}) = 0$ into account yields

$$\begin{aligned} 0 = & \hat{H}_t(\tilde{c})\ddot{t}(\bar{s}) + \hat{H}_u(\tilde{c})\ddot{u}(\bar{s}) + \hat{H}_v(\tilde{c})\ddot{v}(\bar{s}) + \\ & + \hat{H}_{uu}(\tilde{c}) [\dot{u}(\bar{s}), \dot{u}(\bar{s})] + \hat{H}_{vv}(\tilde{c}) [\dot{v}(\bar{s}), \dot{v}(\bar{s})] + 2\hat{H}_{uv}(\tilde{c}) [\dot{u}(\bar{s}), \dot{v}(\bar{s})] \end{aligned} \quad (21)$$

Let $(e_1, e_2)^T$ be in the $\ker \hat{H}'(\tilde{c})^*$, i.e.

$$\hat{H}'(\tilde{c})^* \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \iff \begin{pmatrix} H_t^r(\tilde{c})^* & -H_t^i(\tilde{c})^* \\ H_u^r(\tilde{c})^* & -H_u^i(\tilde{c})^* \\ -H_u^i(\tilde{c})^* & -H_u^r(\tilde{c})^* \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

which can be rewritten as:

$$\begin{cases} H_t^r(\tilde{c})^* e_1 - H_t^i(\tilde{c})^* e_2 = 0 \\ H_u^r(\tilde{c})^* e_1 - H_u^i(\tilde{c})^* e_2 = 0 \\ H_u^i(\tilde{c})^* e_1 + H_u^r(\tilde{c})^* e_2 = 0 \end{cases} \quad (22)$$

Multiplying (21) from the left with $(e_2^*, -e_1^*)$ gives:

$$\begin{aligned} 0 = & (e_2^* H_t^r(\tilde{c}) + e_1^* H_t^i(\tilde{c})) \ddot{t}(\bar{s}) + \\ & + (e_2^* H_u^r(\tilde{c}) + e_1^* H_u^i(\tilde{c})) \ddot{u}(\bar{s}) + (e_2^* H_v^r(\tilde{c}) + e_1^* H_v^i(\tilde{c})) \ddot{v}(\bar{s}) + \\ & + (e_2^* H_{uu}^r(\tilde{c}) + e_1^* H_{uu}^i(\tilde{c})) [\dot{u}(\bar{s}), \dot{u}(\bar{s})] + (e_2^* H_{vv}^r(\tilde{c}) + e_1^* H_{vv}^i(\tilde{c})) [\dot{v}(\bar{s}), \dot{v}(\bar{s})] + \\ & + 2(e_2^* H_{uv}^r(\tilde{c}) + e_1^* H_{uv}^i(\tilde{c})) [\dot{u}(\bar{s}), \dot{v}(\bar{s})] \end{aligned}$$

Using (13), (20), and (22), this simplifies to:

$$\begin{aligned} (e_2^* H_t^r(\tilde{c}) + e_1^* H_t^i(\tilde{c})) \ddot{t}(\bar{s}) &= 2 (e_2^* H_{uu}^i(\tilde{c}) - e_1^* H_{uu}^r(\tilde{c})) [\dot{u}(\bar{s}), \dot{v}(\bar{s})] - \\ &\quad - (e_2^* H_{uu}^r(\tilde{c}) + e_1^* H_{uu}^i(\tilde{c})) \{[\dot{u}(\bar{s}), \dot{u}(\bar{s})] - [\dot{v}(\bar{s}), \dot{v}(\bar{s})]\} \end{aligned}$$

Substituting each of

$$\begin{aligned} \dot{\hat{c}}_1(\bar{s}) &= (0, \dot{u}_1(\bar{s}), \dot{v}_1(\bar{s})) \\ \dot{\hat{c}}_2(\bar{s}) &= \pm (0, \dot{v}_1(\bar{s}), -\dot{u}_1(\bar{s})) \end{aligned}$$

into the above identity, we obtain

$$\begin{aligned} (e_2^* H_t^r(\tilde{c}) + e_1^* H_t^i(\tilde{c})) \ddot{t}_1(\bar{s}) &= 2 (e_2^* H_{uu}^i(\tilde{c}) - e_1^* H_{uu}^r(\tilde{c})) [\dot{u}_1(\bar{s}), \dot{v}_1(\bar{s})] - \\ &\quad - (e_2^* H_{uu}^r(\tilde{c}) + e_1^* H_{uu}^i(\tilde{c})) \{[\dot{u}_1(\bar{s}), \dot{u}_1(\bar{s})] - [\dot{v}_1(\bar{s}), \dot{v}_1(\bar{s})]\} \end{aligned} \tag{23}$$

$$\begin{aligned} (e_2^* H_t^r(\tilde{c}) + e_1^* H_t^i(\tilde{c})) \ddot{t}_2(\bar{s}) &= 2 (e_2^* H_{uu}^i(\tilde{c}) - e_1^* H_{uu}^r(\tilde{c})) [\dot{v}_1(\bar{s}), -\dot{u}_1(\bar{s})] - \\ &\quad - (e_2^* H_{uu}^r(\tilde{c}) + e_1^* H_{uu}^i(\tilde{c})) \{[\dot{v}_1(\bar{s}), \dot{v}_1(\bar{s})] - [\dot{u}_1(\bar{s}), \dot{u}_1(\bar{s})]\} \end{aligned} \tag{24}$$

respectively. One can easily see that the right hand sides of (23) and (24) are equal and of opposite signs, and since $e_2^* H_t^r(\tilde{c}) + e_1^* H_t^i(\tilde{c}) \neq 0$ (see the last part of the proof of the previous theorem), we conclude that $\ddot{t}_1(s) = -\ddot{t}_2(s)$. \square

For a real solution curve of $\hat{H}^{-1}(0)$, we can relax a part of the hypothesis of the theorem 1.3.7 and obtain the following result found also in subchapter 11.8 from [11]

Corollary 1.3.9. *Let $s \mapsto \hat{c}(s) = (\lambda(s), u(s), 0)$ be a "real" solution curve of $\hat{H}^{-1}(0)$ such that the point $(\lambda(\bar{s}), u(\bar{s}))$ is a regular point of the real homotopy H . Suppose $(\lambda(\bar{s}), u(\bar{s}))$ is a simple turning point of the equation $H = 0$, i.e. $\dot{\lambda}(\bar{s}) = 0$ and $\ddot{\lambda}(\bar{s}) \neq 0$. Then $\hat{c}(\bar{s})$ is a simple bifurcation point of the equation $\hat{H} = 0$.*

1.4 Possible Tracked Paths and the ‘End Game’ of Tracking

No matter what method we use to track the solutions of the homotopy H_N from $t = 1$ to $t = 0$, we must land exactly on $t = 0$ because only here, the incompatibility between the old boundary conditions and the new ones disappears (see Remark 1.2.2 for more details). In this sense, the last step of the tracking method (landing exactly on $t = 0$) plays an important role in our implementation and was called *the ‘End Game’ of Tracking*.

Here are the two types of ‘End Game’ we used in our research.

- i) As we track a solution, the homotopy parameter t will decrease from 1 toward 0. When t is very close to 0, we will chose the last steplength such that t lands directly on 0. Using this method, t will always stay in the interval $[0, 1]$.
- ii) The second idea is to track a solution until t passes 0. We stop when t becomes for the first time negative and choose now an opposite steplength such that t lands directly on 0.

It is important to remark that the tracker from HomLab software package used by the authors in [1] is based on the first idea. We also used ‘end game’ (i) for tracking our solutions with continuation in t , applying the gamma-trick when necessary. We tried to use this ‘end game’ also for tracking using arclength continuation, but a major inconvenience arose when turning points of tracked solutions occurred near $t = 0$. In this case, we could end up at $t = 0$ with a solution that had already been obtained from a different starting solution at $t = 1$, by jumping to a different solution branch that is not connected to our actual one. We call this losing solutions. See section 2.7.1 for an example. This problem holds also for [1], but was not observed there. Another problem not observed in [1] is also related to the turning points near $t = 1$ where branches must be switched, but as the authors said, there is a zero probability that turning points appear when the gamma-trick is used.

The ‘disadvantage’ of the second idea according to which we will need an extra last step to land on $t = 0$ is insignificant in comparison with the following advantage: any turning

point near $t = 0$ will be seen and the switch to the right branch as suggested in section 1.3 can be done in the tracking process. This way, we no longer lose solutions as we did using the previous idea.

Therefore, we suggest using the ‘end game’ (ii) combined with tracking by arclength continuation as presented before in section 1.3.

Possible paths that we might encounter during our tracking are presented in the figure below.

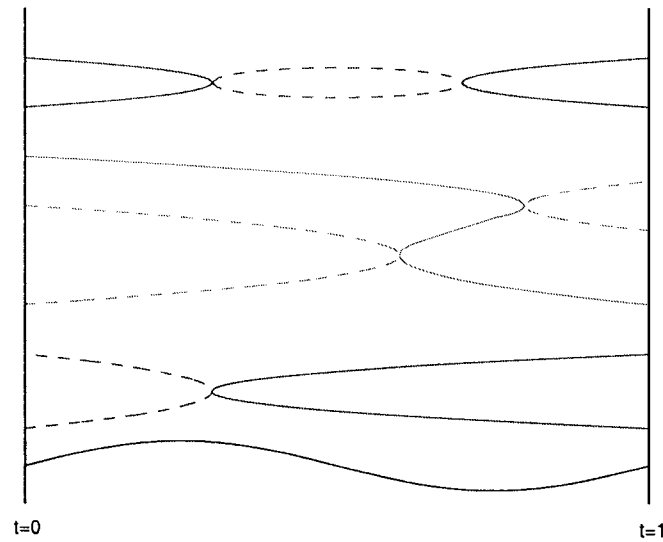


Figure 2: Possible paths in our tracking.

The easiest path to track is the one with no turning points going directly from $t = 1$ to $t = 0$.

A different type of path is the one starting and ending back at $t = 1$ because of a turning point somewhere between 1 and 0. In this case, by the theory presented in section 1.3, there is another path with opposite curvature having the same bifurcation point. This new path might reach $t = 0$ directly or continue to have turning points. In any case, we should be able to switch branches as in section 1.3 and reach our goal, $t = 0$.

A third type of path is the one starting at $t = 1$ and ending at $t = 0$, having two or more turning points, and for which we do not need to switch branches to reach our goal.

Of course, from a numerical point of view, the hard cases are when the turning points are near $t = 1$ or $t = 0$.

It is also very important to remark that we do not get solutions going to infinity as in the picture below because of the existence of apriori estimates for certain kinds of elliptic problems (see [15] for more details).

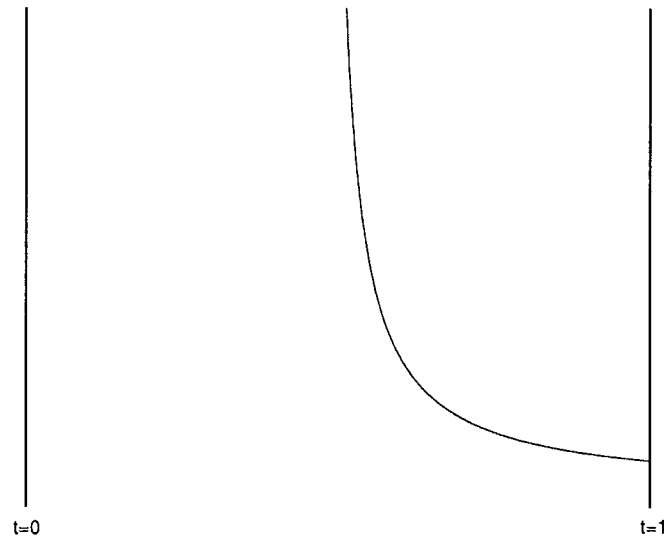


Figure 3: Path going to infinity.

CHAPTER II

HOMOTOPY CONTINUATION FOR 2-D PARTIAL DIFFERENTIAL EQUATIONS

The method of finding all the solutions of second order ordinary differential equations presented in the previous chapter is now generalized to a class of nonlinear second order semilinear elliptic boundary value problems with homogeneous boundary conditions on a rectangular domain, but still with polynomial nonlinearities. Most of the difficulties met for the 1-D case are now transmitted to the new type of problems. Other difficulties that appear due to the generalization to a higher dimension are also described in the first section of this chapter.

Using the geometry of a rectangular box-interval, we were able to build the homotopy function associated to the discretization of the Laplace operator as a combination of some sparse matrices and vectors. All the calculations were performed for the case when a new mesh point is introduced in a row. For the other case (new mesh point introduced on a column), the calculations are similar and therefore we did not present them in this thesis. But both of these cases were implemented since we need to alternate introducing mesh points on rows and columns. The geometry of the moving mesh-grid is presented in the fifth section of this chapter.

The section 2.6 describes the way we refine a solution to a coarser grid once we numerically found it on a crude mesh. Some numerical results for different 2-D problems are presented in the section 2.7.

2.1 *Difficulties for 1-D ODEs and 2-D PDEs*

We now turn to the question of whether it's feasible to solve similar problems, but in 2-D or higher dimensions. In particular, we will concentrate on finding all the solutions for a problem of the form

$$\begin{aligned}\Delta u &= f(\lambda, x, y, u, u_x, u_y) && \text{on } \Omega \subset \mathbb{R}^2 \\ u|_{\partial\Omega} &= g.\end{aligned}$$

In the next chapters we will develop a theory and a toolbox to solve this problem numerically. But first, let's explain some difficulties we met in the 1-D case.

Generalizing the right hand side to non-polynomial functions

The algorithm described in 1.2 works well for the case when the function f is a polynomial in u , because in this case, (8) becomes a polynomial equation (see (1.2)) for which one knows how to find *all* the solutions. There is no method to find all solutions if the nonlinearity f is not a polynomial anymore. This difficulty will also apply for the 2-D case and higher dimensions.

Exponential growth for the number of the solutions

Even in the case of a polynomial nonlinearity, the number of solutions grows exponentially with the number of interior points. The algorithm will stop when we have enough interior points or when we see that the number of real solutions stabilizes or grows without bound. In all the 1-D problems that we have examined, the number of real solutions stabilized or grew without bound. Once this was accomplished, we were able to take these solutions with 6-7 interior points and refine them using a simple numerical method (Newton iteration for example). See the section 2.6 for more information. So, now imagine, if 6-7 interior points for a 1-D interval will be satisfactory (providing that the number of real solutions stabilizes or grows without bound), then for a 2-D problem on a rectangular domain we would be satisfied with 36-49 interior points. Clearly so many mesh points will lead to a

great number of solutions for the homotopy and this will require stricter tests for filtering out unwanted solutions. For the 1-D case, knowing some properties about f would help us creating some filters, for example, maybe it can be shown that the solution needs to have some symmetries. But in 2-D or higher dimensions, this difficulty becomes greater.

No solutions for the homotopy function

Another difficulty that was met in 1-D is the fact that we would like to know if we can find *all* the solutions also for the case in which f is not a polynomial in u . Lots of problems arising from engineering, chemistry, biology or other areas do not involve polynomial nonlinearities. In order to find all the solutions even if f is not a polynomial in u , we need a method that can find all solutions for (8). So, if we know some properties and methods to solve (8) for all the solutions (keep in mind that (8) will be an equation in which some parameters will change every time we come with new interior points) then we are in good shape. But even in this case, we might have problems: suppose we know how to find all the real solution for (8) for the general case; what if the equation (8) we solve for small N (interior points) does not have real solutions (see section 4.1)? How can we continue with our algorithm, since we don't have any starting solution for our homotopy? Does it mean that the problem we are solving does not have any solution? No, it does not! At a first glance, it seems like we cannot go further with our homotopy in this case. It is not really like that. Remember that even for the polynomial case in 1-D, the idea of throwing away the complex solutions could be a mistake, because there might be real solutions coming up later in the process (after adding more interior points) that are born from complex ones. This scenario was noticed in [1] and also holds our case. It is possible to have no real solutions for small N for (8), but to have complex solutions that usually give birth to real ones once we add more interior points. Hence, it is not sufficient to know how to solve (8) only for the real solutions, you need to know how to solve it also for the complex ones! A better understating of this difficulty will be given in section 4.1. A solution for this problem

will come from exclusion algorithms, but these algorithms help find all the real roots of a system from a given box-interval, not from all \mathbb{R}^n . This is still very helpful since, for the problems from engineering or other fields, we are usually looking for solutions in a specific box-interval.

Turning points

Another difficulty which was not really met in the 1-D problems we looked at was the continuation method we used to find the roots of the homotopy. As the authors in [1] we used numerical continuation in t (the homotopy parameter), since we did not really meet cases of turning points. But in 2-D, we often met cases of turning points for continuation in t , hence we needed to use the gamma-trick or arclength numerical continuation. We also had to be very careful with the steplength since we had cases of turning points very close to $t = 1$ or $t = 0$ (see section 2.7.1 for example).

2.2 Details on the 2-D grid for $\Delta u = f(\lambda, x, y, u, u_x, u_y)$

The number of solutions and indeed the very existence of solutions to general PDE's is of great interest. Various particular cases of PDE's have been studied from a theoretical point of view, but there is no general result about the number of solutions. In this chapter, numerical continuation will be used also to find all the solutions for a problem of the form

$$\begin{aligned} \Delta u &= f(\lambda, x, y, u, u_x, u_y) & \text{on } \Omega \subset \mathbb{R}^2 \\ u|_{\partial\Omega} &= g. \end{aligned} \tag{25}$$

Using a standard central difference approximation, with a uniform mesh with N interior points, the discretization of (25) will take the form of the system

$$\mathcal{D}_N : \quad A_{xx}\vec{u} + \vec{b}_{xx} + A_{yy}\vec{u} + \vec{b}_{yy} = \vec{f} \tag{26}$$

where

- $\vec{u} = (u_1, u_2, \dots, u_N)^T$ is the discretized solution vector of the unknowns,
- $A = A_{xx} + A_{yy}$ represents what is called in the literature *the stiffness matrix* (A_{xx} comes from $\frac{\partial^2 u}{\partial x^2}$ and A_{yy} comes from $\frac{\partial^2 u}{\partial y^2}$),
- the vectors \vec{b}_{xx} and \vec{b}_{yy} arise from the boundary values of the discretization of (25)(see the next numerical example for a better understanding).

Here, the discretization for $\frac{\partial^2 u}{\partial x^2}$ with incorporating the left and right boundary conditions is $A_{xx}\vec{u} + \vec{b}_{xx}$. The discretization for $\frac{\partial^2 u}{\partial y^2}$ with incorporating the upper and lower boundary conditions is $A_{yy}\vec{u} + \vec{b}_{yy}$.

Note: These notations A_{xx} , A_{yy} , \vec{b}_{xx} , \vec{b}_{yy} might appear unconventional, but here are two reasons for using them: they give the idea where they come from (for instance, A_{xx} and \vec{b}_{xx} come from discretizing $\frac{\partial^2 u}{\partial x^2}$ with incorporating the boundary conditions) and each of them has a nice sparse form (we will see it later).

Let's focus now on studying the problem (25) for the case where f is a polynomial of u and $\Omega = [a, b] \times [c, d]$. As in 1-D, we will continue introducing *one* point at a time for the 2-D case also.

It is unnecessary to introduce a row or a column of p points at a time instead of one point and there are a few reasons to justify this. Introducing a row or a column of p points would require solving a system of p polynomial equations to find the starting solutions for our homotopy. If the degree of f is d for example, then each time we add a new point we have to track d new solutions of the homotopy as t (the homotopy parameter) goes from 1 to 0. After introducing p points (each one at a time), we end up tracking down around d^p solutions. If we introduce a row or a column of p points at once instead, then solving that system of p equations would give us the same number of solutions to track, and therefore adding the task of solving such a system is not justified. Moreover, if f is not a polynomial function anymore, then solving such a system would become even a harder task.

Another reason is related to filters. After introducing a row or column of p points at a time and tracking all d^p solutions, we can apply some filters (symmetry filters if available for example) to eliminate some unwanted solutions; but we still had to track all d^p solutions. If instead we are introducing one point at a time, we might be able to still apply some filters (not all the previous symmetry filters might be available) and this way, after introducing p points, but each one at a time, we may end up tracking less than d^p solutions. This will help us saving lots of calculation and time for any big number of interior points.

2.3 Introducing a point on the l^{th} row. Approximating the derivatives.

Let's rewrite the BVP (Boundary Value Problem) (25) as:

$$\Delta u = f(\lambda, x, y, u, u_x, u_y) \quad \text{on } \Omega = [a, b] \times [c, d] \subset \mathbb{R}^2 \quad (27)$$

with BC's $u|_{\partial\Omega}$:

$$\begin{aligned} u|_{x=a} &= u_a(y) \\ u|_{x=b} &= u_b(y) \\ u|_{y=c} &= u_c(x) \\ u|_{y=d} &= u_d(x) \end{aligned} \quad (28)$$

The main obstacle to be overcome when introducing extra points one at a time is now to approximate the Laplacian on non-regular finite difference meshes. This is achieved by appropriate application of linear interpolation.

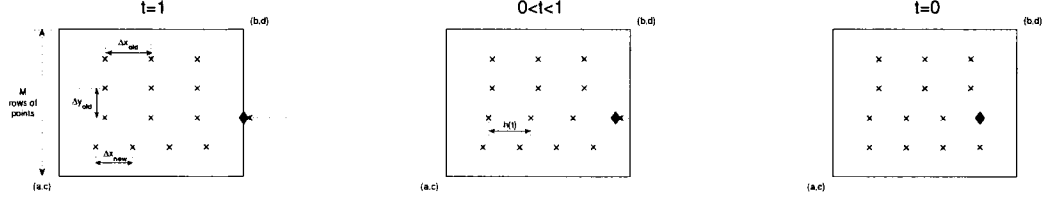


Figure 4: Introducing a new point (♦) on the l^{th} row: the picture at the initial step $t = 1$ of the homotopy (*left picture*), the picture at an intermediate step $0 < t < 1$ of the homotopy (*central picture*), and the picture at the final step $t = 0$ of the homotopy (*right picture*).

General Case: $0 < t < 1$

When we introduce a new point on the l^{th} column (\blacklozenge) from the right, all the other points on the l^{th} row shift leftward as in the figure from below. Remark that rows $1, 2, \dots, l$ have $N + 1$ points each, and rows $l + 1, l + 2, \dots, M$ have N points each.

- $\Delta x_{old} = \frac{b-a}{N+1}$
- $\Delta x_{new} = \frac{b-a}{N+2}$
- $\Delta y_{old} = \frac{d-c}{M+1}$
- $h(t) = t \left(\frac{b-a}{N+1} \right) + (1-t) \left(\frac{b-a}{N+2} \right)$
- $d(t) = (b-a) - (N+1)h(t) = \dots = (1-t) \frac{b-a}{N+2}$

For the following particular picture, our discretized solution \vec{u} of the system (26) will be

$$\vec{u} = [u_{1,1}, u_{2,1}, u_{3,1}, u_{4,1}, u_{1,2}, u_{2,2}, u_{3,2}, u_{4,2}, \dots, u_{1,5}, u_{2,5}, u_{3,5}]^T_{18 \times 1}.$$

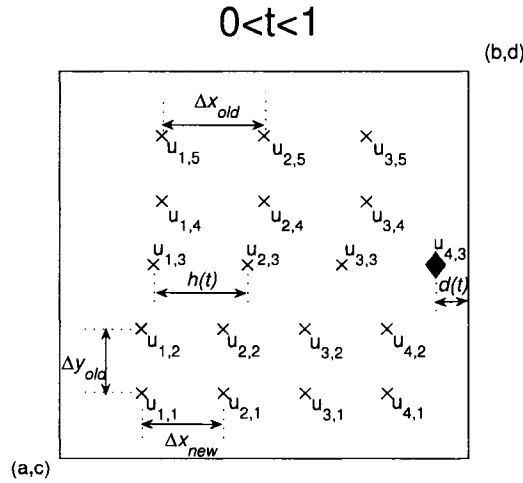


Figure 5: Introducing a new point (\blacklozenge) on the l^{th} row: the general case of the homotopy: $0 < t < 1$. Approximating the Laplacian at all the points, except the ones on the rows $l - 1, l$, and $l + 1$.

Remark 2.3.0.1. Notice the way we order the unknowns $u_{i,j}$ inside of the vector \vec{u} . When a new point will be introduced on a **column** (from above for example) instead of a **row**, the unknowns $u_{i,j}$ inside of the vector \vec{u} will be ordered differently (to keep the sparsity structure of the stiffness matrix A). In this case, the vector \vec{u} will have the form

$$\vec{u} = [u_{1,1}, u_{1,2}, u_{1,3}, u_{1,4}, u_{1,5}, u_{2,1}, u_{2,2}, \dots]^T.$$

2.3.1 Approximating the Laplacian at all the interior points except the ones on the rows $l - 1, l$, and $l + 1$

We can easily approximate the Laplacian at all the points which are not on the rows $l - 1, l$, and $l + 1$ by

- Formula for $(u_{xx})_{j,k}$

$$(u_{xx})_{j,k} \approx \frac{u_{j-1,k} - 2u_{j,k} + u_{j+1,k}}{\Delta x^2}, \quad \forall j = 1, \dots, N \text{ (or } N + 1), \quad \forall k = 1, \dots, M,$$

$k \neq l$, where

$$\Delta x = \begin{cases} \Delta x_{new} & \text{if } 1 \leq k \leq l - 1 \\ \Delta x_{old} & \text{if } l + 1 \leq k \leq M \end{cases}.$$

- Formula for $(u_{yy})_{j,k}$

$$(u_{yy})_{j,k} \approx \frac{u_{j,k-1} - 2u_{j,k} + u_{j,k+1}}{\Delta y^2}, \quad \forall j = 1, \dots, N \text{ (or } N + 1), \quad \forall k = 1, \dots, M,$$

$k \notin \{l - 1, l, l + 1\}$, where

$$\Delta y = \Delta y_{old}.$$

2.3.2 Approximating the Laplacian at the interior points on the $(l-1)^{th}$ row

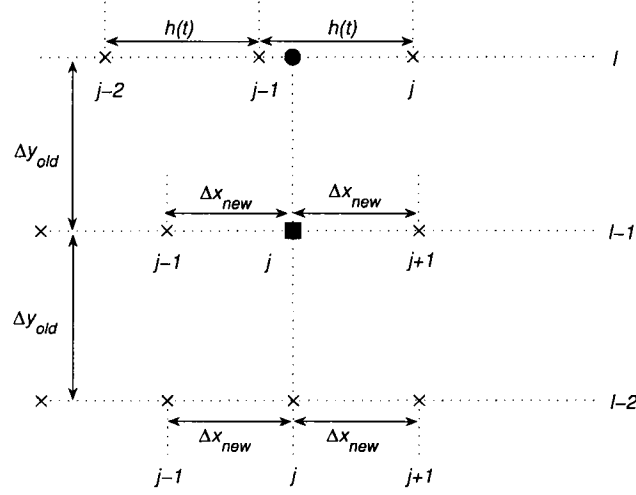


Figure 6: Introducing a new point (◆) on the l^{th} row. Approximating the Laplacian at the points on the $(l-1)^{th}$ row.

On the $(l-1)^{th}$ row we have the following.

- Formula for $(u_{xx})_{j,l-1}$

$$(u_{xx})_{j,l-1} \approx \frac{u_{j-1,l-1} - 2u_{j,l-1} + u_{j+1,l-1}}{\Delta x^2}, \quad \forall j = 1, \dots, N+1, \quad \text{where}$$

$$\Delta x = \Delta x_{new}.$$

- Formula for $(u_{yy})_{j,l-1}$

$$(u_{yy})_{j,l-1} \approx \frac{u_{j,l-2} - 2u_{j,l-1} + u_{\bullet}}{\Delta y^2}, \quad \forall j = 1, \dots, N+1, \quad \text{where}$$

$$\Delta y = \Delta y_{old}$$

$$u_{\bullet} \approx \tilde{u}_{\bullet} := u_{j-1,l} + r_{\bullet} \cdot (u_{j,l} - u_{j+1,l}), \quad \text{where}$$

$$r_{\bullet} = \frac{j\Delta x_{new} - (j-1)h(t)}{h(t)} = \dots = 1 - j \frac{t}{t+N+1}.$$

Hence,

$$(u_{yy})_{j,l-1} \approx \frac{(1-r_{\bullet})u_{j-1,l} + u_{j,l-2} - 2u_{j,l-1} + r_{\bullet}u_{j,l}}{\Delta y_{old}^2}, \quad \forall j = 1, \dots, N+1.$$

2.3.3 Approximating the Laplacian at the interior points on the l^{th} row, except at the new point (◆)

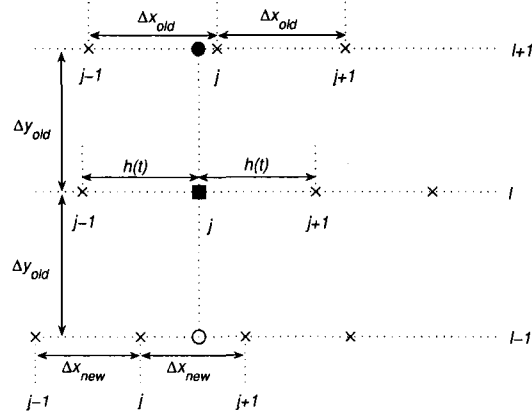


Figure 7: Introducing a new point (◆) on the l^{th} row. Approximating the Laplacian at the points on the l^{th} row *except at the new point* (◆).

On the l^{th} row, **except for the new point** (◆), we have the following.

- Formula for $(u_{xx})_{j,l}$

$$(u_{xx})_{j,l} \approx \frac{u_{j-1,l} - 2u_{j,l} + u_{j+1,l}}{h(t)^2}, \quad \forall j = 1, \dots, N.$$

- Formula for $(u_{yy})_{j,l}$

$$(u_{yy})_{j,l} \approx \frac{u_{\circ} - 2u_{j,l} + u_{\bullet}}{\Delta y^2}, \quad \forall j = 1, \dots, N, \quad \text{where}$$

$$\Delta y = \Delta y_{old}$$

$$u_{\circ} \approx \tilde{u}_{\circ} := u_{j,l-1} + r_{\circ} \cdot (u_{j+1,l-1} - u_{j,l-1}), \quad \text{where}$$

$$r_{\circ} = \frac{j \cdot h(t) - j \Delta x_{new}}{\Delta x_{new}} = \dots = j \frac{t}{N+1}$$

$$u_{\bullet} \approx \tilde{u}_{\bullet} := u_{j-1,l+1} + r_{\bullet} \cdot (u_{j,l+1} - u_{j-1,l+1}), \quad \text{where}$$

$$r_{\bullet} = \frac{j \cdot h(t) - (j-1) \Delta x_{old}}{\Delta x_{old}} = \dots = 1 - j \frac{1-t}{N+2}.$$

Hence,

$$(u_{yy})_{j,l} \approx \frac{(1-r_{\bullet})u_{j-1,l+1} + (1-r_{\circ})u_{j,l-1} - 2u_{j,l} + r_{\bullet}u_{j,l+1} + r_{\circ}u_{j+1,l-1}}{\Delta y_{old}^2}, \quad \forall j = 1, \dots, N.$$

2.3.4 Approximating the Laplacian at the new point (♦) introduced on the l^{th} row

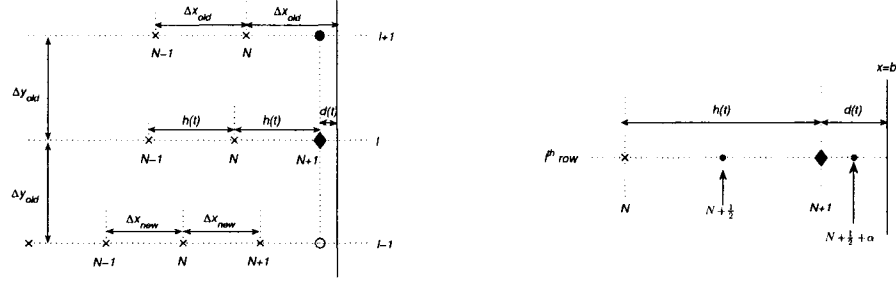


Figure 8: Introducing a new point (♦) on the l^{th} row. Approximating the Laplacian at the new point (♦).

On the l^{th} row, for the new point (♦), we have the following.

- First, let's calculate the distance from the new point (♦) to the boundary line $x = b$:

$$d(t) = (b - a) - (N + 1)h(t) = \dots = (1 - t) \frac{b-a}{N+2}.$$

Observe that $d(t) = r(t)h(t)$, where $r(t) = \frac{(1-t)(N+1)}{t+N+1}$.

- Formula for $(u_{xx})_{N+1,l}$

$$(u_{xx})_{N+1,l} \approx \frac{(u_x)_{N+\frac{1}{2},l} - (u_x)_{N+1+\frac{\alpha}{2},l}}{\frac{h(t)}{2} + \frac{d(t)}{2}} \approx \frac{\frac{u_{N,l} - u_{N+1,l}}{h(t)} - \frac{u_{N+1,l} - u_b(y_l)}{d(t)}}{\frac{h(t)}{2} + \frac{d(t)}{2}} = \dots$$

Hence, $(u_{xx})_{N+1,l} \approx \frac{r(t)u_{N,l} - [1+r(t)]u_{N+1,l} + u_b(c+l \cdot \Delta y_{old})}{\frac{h(t)^2 r(t)[1+r(t)]}{2}}$, where $r(t) = \frac{(1-t)(N+1)}{t+N+1}$.

- Formula for $(u_{yy})_{N+1,l}$

$$(u_{yy})_{N+1,l} \approx \frac{u_o - 2u_{N+1,l} + u_{\bullet}}{\Delta y^2}, \text{ where } \Delta y = \Delta y_{old} \text{ and}$$

$$u_o \approx \tilde{u}_o := u_{N+1,l-1} + r_o \cdot (u_b(y_{l-1}) - u_{N+1,l-1}), \quad \text{where}$$

$$r_o = \frac{(N+1)h(t) - (N+1)\Delta x_{new}}{\Delta x_{new}} = \dots = t$$

$$u_{\bullet} \approx \tilde{u}_{\bullet} := u_{N,l+1} + r_{\bullet} \cdot (u_b(y_{l+1}) - u_{N,l+1}), \quad \text{where}$$

$$r_{\bullet} = \frac{(N+1)h(t) - N\Delta x_{old}}{\Delta x_{old}} = \dots = t + (1-t) \frac{1}{N+2}.$$

Hence, $(u_{yy})_{N+1,l} \approx \frac{(1-r_{\bullet})u_{N,l+1} + (1-r_o)u_{N+1,l-1} - 2u_{N+1,l} + r_{\bullet}u_b(y_{l+1}) + r_o u_b(y_{l-1})}{\Delta y_{old}^2}.$

2.3.5 Approximating the Laplacian at the interior points on the $(l+1)^{th}$ row

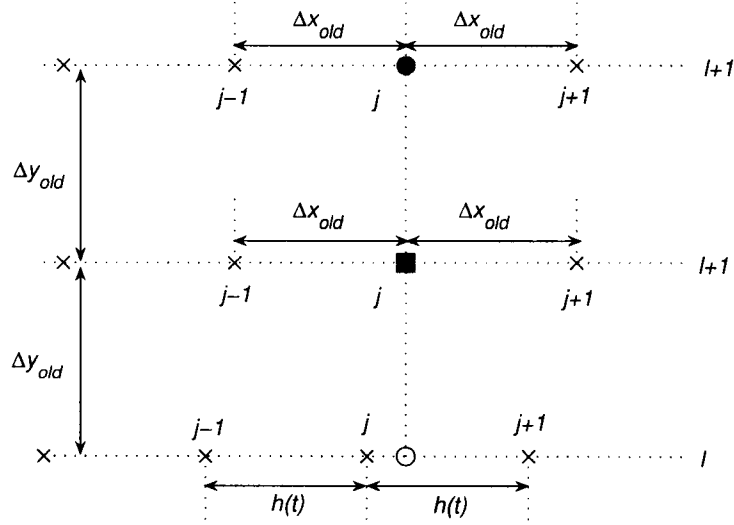


Figure 9: Introducing a new point (♦) on the l^{th} row. Approximating the Laplacian at the points on the $(l+1)^{th}$ row.

On the $(l+1)^{th}$ row we have the following.

- Formula for $(u_{xx})_{j,l+1}$

$$(u_{xx})_{j,l+1} \approx \frac{u_{j-1,l+1} - 2u_{j,l+1} + u_{j+1,l+1}}{\Delta x^2}, \quad \forall j = 1, \dots, N, \quad \text{where } \Delta x = \Delta x_{old}.$$

- Formula for $(u_{yy})_{j,l+1}$

$$(u_{yy})_{j,l+1} \approx \frac{u_o - 2u_{j,l+1} + u_{j,l+2}}{\Delta y^2}, \quad \forall j = 1, \dots, N, \quad \text{where}$$

$$\Delta y = \Delta y_{old}$$

$$u_o \approx \tilde{u}_o := u_{j,l} + r_o \cdot (u_{j+1,l} - u_{j,l}), \quad \text{where}$$

$$r_o = \frac{j\Delta x_{old} - jh(t)}{h(t)} = \dots = j \frac{1-t}{t+N+1}.$$

Hence,

$$(u_{yy})_{j,l+1} \approx \frac{(1-r_o)u_{j,l} - 2u_{j,l+1} + u_{j,l+2} + r_o u_{j+1,l}}{\Delta y_{old}^2}, \quad \forall j = 1, \dots, N.$$

2.3.6 The discretized equations for $\Delta u = f(\lambda, x, y, u, u_x, u_y)$ necessary to build the homotopy.

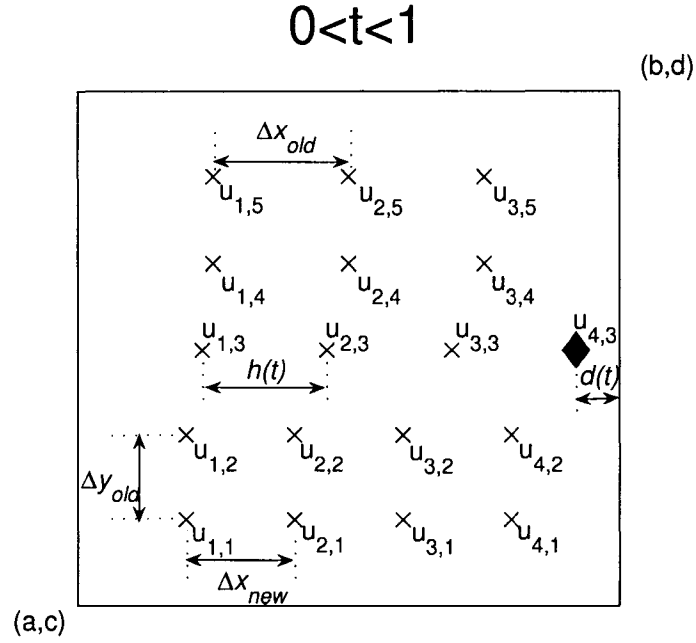


Figure 10: Introducing a new point (♦) on the l^{th} row: the general case of the homotopy, $0 < t < 1$.

We can now easily write for this particular case the equations of the discretized system which will help us later in building the homotopy for the general case

$$\delta_x^2 u_{i,j} + \delta_y^2 u_{i,j} = f_{i,j} \quad , \quad (29)$$

where

$$\begin{aligned}
\delta_x^2 u_{1,1} &= \frac{1}{\Delta x_{new}^2} (u_a(y_1) - 2u_{1,1} + u_{2,1}) \\
\delta_x^2 u_{2,1} &= \frac{1}{\Delta x_{new}^2} (u_{1,1} - 2u_{2,1} + u_{3,1}) \\
\delta_x^2 u_{3,1} &= \frac{1}{\Delta x_{new}^2} (u_{2,1} - 2u_{3,1} + u_{4,1}) \\
\delta_x^2 u_{4,1} &= \frac{1}{\Delta x_{new}^2} (u_{3,1} - 2u_{4,1} + u_b(y_1)) \\
\delta_y^2 u_{1,1} &= \frac{1}{\Delta y_{old}^2} [u_c(x_{1new}) - 2u_{1,1} + u_{1,2}] \\
\delta_y^2 u_{2,1} &= \frac{1}{\Delta y_{old}^2} [u_c(x_{2new}) - 2u_{2,1} + u_{2,2}] \\
\delta_y^2 u_{3,1} &= \frac{1}{\Delta y_{old}^2} [u_c(x_{3new}) - 2u_{3,1} + u_{3,2}] \\
\delta_x^2 u_{4,1} &= \frac{1}{\Delta y_{old}^2} [u_c(x_{4new}) - 2u_{4,1} + u_{4,2}] \\
\delta_x^2 u_{1,2} &= \frac{1}{\Delta x_{new}^2} (u_a(y_2) - 2u_{1,2} + u_{2,2}) \\
\delta_x^2 u_{2,2} &= \frac{1}{\Delta x_{new}^2} (u_{1,2} - 2u_{2,2} + u_{3,2}) \\
\delta_x^2 u_{3,2} &= \frac{1}{\Delta x_{new}^2} (u_{2,2} - 2u_{3,2} + u_{4,2}) \\
\delta_x^2 u_{4,2} &= \frac{1}{\Delta x_{new}^2} (u_{3,2} - 2u_{4,2} + u_b(y_2)) \\
\delta_y^2 u_{1,2} &= \frac{1}{\Delta y_{old}^2} [(1 - r_\bullet) u_a(y_3) + u_{1,1} - 2u_{1,2} + r_\bullet u_{1,3}], \quad \text{where } r_\bullet = 1 - 1 \frac{t}{t+N+1} \\
\delta_y^2 u_{2,2} &= \frac{1}{\Delta y_{old}^2} [(1 - r_\bullet) u_{1,3} + u_{2,1} - 2u_{2,2} + r_\bullet u_{2,3}], \quad \text{where } r_\bullet = 1 - 2 \frac{t}{t+N+1} \\
\delta_y^2 u_{3,2} &= \frac{1}{\Delta y_{old}^2} [(1 - r_\bullet) u_{2,3} + u_{3,1} - 2u_{3,2} + r_\bullet u_{3,3}], \quad \text{where } r_\bullet = 1 - 3 \frac{t}{t+N+1} \\
\delta_y^2 u_{4,2} &= \frac{1}{\Delta y_{old}^2} [(1 - r_\bullet) u_{3,3} + u_{4,1} - 2u_{4,2} + r_\bullet u_{4,3}], \quad \text{where } r_\bullet = 1 - 4 \frac{t}{t+N+1} \\
\delta_x^2 u_{1,3} &= \frac{1}{h(t)^2} (u_a(y_3) - 2u_{1,3} + u_{2,3}) \\
\delta_x^2 u_{2,3} &= \frac{1}{h(t)^2} (u_{1,3} - 2u_{2,3} + u_{3,3}) \\
\delta_x^2 u_{3,3} &= \frac{1}{h(t)^2} (u_{2,3} - 2u_{3,3} + u_{4,3}) \\
\delta_y^2 u_{1,3} &= \frac{1}{\Delta y_{old}^2} [(1 - r_\bullet) u_a(y_4) + (1 - r_\circ) u_{1,2} - 2u_{1,3} + r_\bullet u_{1,4} + r_\circ u_{2,2}], \\
&\quad \text{where } r_\bullet = 1 - 1 \frac{1-t}{N+2}, \quad r_\circ = 1 \frac{t}{N+1} \\
\delta_y^2 u_{2,3} &= \frac{1}{\Delta y_{old}^2} [(1 - r_\bullet) u_{1,4} + (1 - r_\circ) u_{2,2} - 2u_{2,3} + r_\bullet u_{2,4} + r_\circ u_{3,2}], \\
&\quad \text{where } r_\bullet = 1 - 2 \frac{1-t}{N+2}, \quad r_\circ = 2 \frac{t}{N+1} \\
\delta_y^2 u_{3,3} &= \frac{1}{\Delta y_{old}^2} [(1 - r_\bullet) u_{2,4} + (1 - r_\circ) u_{3,2} - 2u_{3,3} + r_\bullet u_{3,4} + r_\circ u_{4,2}], \\
&\quad \text{where } r_\bullet = 1 - 3 \frac{1-t}{N+2}, \quad r_\circ = 3 \frac{t}{N+1}
\end{aligned}$$

$$\begin{aligned}\delta_x^2 u_{4,3} &= \frac{1}{h(t)^2} \left(\frac{2}{1+r(t)} u_{3,3} - \frac{2}{r(t)} u_{4,3} + \frac{2}{r(t)(1+r(t))} u_b(y_3) \right) \\ \delta_y^2 u_{4,3} &= \frac{1}{\Delta y_{old}^2} [(1-r_\bullet) u_{3,4} + (1-r_\circ) u_{4,2} - 2u_{4,3} + r_\bullet u_b(y_4) + r_\circ u_b(y_2)], \\ &\text{where } r(t) = \frac{(1-t)(N+1)}{t+N+1}, \quad r_\bullet = t + \frac{1-t}{N+2}, \quad r_\circ = t\end{aligned}$$

$$\begin{aligned}\delta_x^2 u_{1,4} &= \frac{1}{\Delta x_{old}^2} (u_a(y_4) - 2u_{1,4} + u_{2,4}) \\ \delta_x^2 u_{2,4} &= \frac{1}{\Delta x_{old}^2} (u_{1,4} - 2u_{2,4} + u_{3,4}) \\ \delta_x^2 u_{3,4} &= \frac{1}{\Delta x_{old}^2} (u_{2,4} - 2u_{3,4} + u_b(y_4)) \\ \delta_y^2 u_{1,4} &= \frac{1}{\Delta y_{old}^2} [(1-r_\circ) u_{1,3} - 2u_{1,4} + u_{1,5} + ru_{2,3}], \quad \text{where } r_\circ = 1 \frac{1-t}{t+N+1} \\ \delta_y^2 u_{2,4} &= \frac{1}{\Delta y_{old}^2} [(1-r_\circ) u_{2,3} - 2u_{2,4} + u_{2,5} + ru_{3,3}], \quad \text{where } r_\circ = 2 \frac{1-t}{t+N+1} \\ \delta_y^2 u_{3,4} &= \frac{1}{\Delta y_{old}^2} [(1-r_\circ) u_{3,3} - 2u_{3,4} + u_{3,5} + ru_{4,3}], \quad \text{where } r_\circ = 3 \frac{1-t}{t+N+1} \\ \delta_x^2 u_{1,5} &= \frac{1}{\Delta x_{old}^2} (u_a(y_5) - 2u_{1,5} + u_{2,5}) \\ \delta_x^2 u_{2,5} &= \frac{1}{\Delta x_{old}^2} (u_{1,5} - 2u_{2,5} + u_{3,5}) \\ \delta_x^2 u_{3,5} &= \frac{1}{\Delta x_{old}^2} (u_{2,5} - 2u_{3,5} + u_b(y_5)) \\ \delta_y^2 u_{1,5} &= \frac{1}{\Delta y_{old}^2} [u_{1,4} - 2u_{1,5} + u_d(x_{1_{old}})] \\ \delta_y^2 u_{2,5} &= \frac{1}{\Delta y_{old}^2} [u_{2,4} - 2u_{2,5} + u_d(x_{2_{old}})] \\ \delta_y^2 u_{3,5} &= \frac{1}{\Delta y_{old}^2} [u_{3,4} - 2u_{3,5} + u_d(x_{3_{old}})]\end{aligned}$$

2.4 The homotopy function for $\Delta u = f(\lambda, x, y, u, u_x, u_y)$

Remember that our BVP

$$\Delta u = f(\lambda, x, y, u, u_x, u_y) \quad \text{on} \quad \Omega = [a, b] \times [c, d] \subset \mathbb{R}^2$$

with $u|_{\partial\Omega}$ given by

$$u|_{x=a} = u_a(y)$$

$$u|_{x=b} = u_b(y)$$

$$u|_{y=c} = u_c(x)$$

$$u|_{y=d} = u_d(x)$$

can be discretized as

$$A_{xx}\vec{u} + \vec{b}_{xx} + A_{yy}\vec{u} + \vec{b}_{yy} = \vec{f} ,$$

where $A = A_{xx} + A_{yy}$ is the **stiffness matrix** and $\vec{b}_{xx}, \vec{b}_{yy}$ are what we will call **boundary vectors**. The corresponding homotopy from (6), but now for this 2-D BVP is:

$$H_{N+1}(\vec{u}, t) := (A_{xx} + A_{yy})\vec{u} + (\vec{b}_{xx} + \vec{b}_{yy}) - \vec{f} \quad (30)$$

where the explicit forms for $A_{xx}, A_{yy}, \vec{b}_{xx}, \vec{b}_{yy}, \vec{f}$ will be given soon. We made a little bit of abuse of notation, since these $A_{xx}, A_{yy}, \vec{b}_{xx}, \vec{b}_{yy}, \vec{f}$ from (30) depend on the homotopy parameter t ; they are indeed the components of the stiffness matrix A , but the one associated to the grid existent at a value $0 \leq t \leq 1$ (see Figure 4). Now, let's introduce the following notations:

- t - the homotopy parameter,
- L - the row where the new point was introduced,
- M - the number of rows of points,
- N - the number of points on each of the rows $L + 1, \dots, M$; hence, rows $1, \dots, L$ each has $N + 1$ points,

- n - the total number of interior points; observe that $n = L(N + 1) + (M - L)N$,
 $\forall t \in [0, 1)$.

With these notations, the two matrices and the two vectors from above will have the following GENERAL SPARSE form.

A_{xx} is a block diagonal matrix of the form ¹

$$A_{xx} = \begin{bmatrix} T_1 & & & & \\ & \ddots & & & \\ & & T_1 & & \\ & & & T_2 & \\ & & & & T_3 \\ & & & & & \ddots \\ & & & & & & T_3 \end{bmatrix}$$

where T_1, T_2, T_3 are tridiagonal square matrices of sizes $N + 1$, $N + 1$, and N , respectively

$$\bullet T_1 = \frac{1}{\Delta x_n^2} \begin{bmatrix} -2 & 1 & & & \\ & 1 & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & -2 \end{bmatrix}$$

$$\bullet T_3 = \frac{1}{\Delta x_o^2} \begin{bmatrix} -2 & 1 & & & \\ & 1 & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & -2 \end{bmatrix}$$

$$\bullet T_2 = \frac{1}{h(t)^2} \begin{bmatrix} -2 & 1 & & & \\ & 1 & -2 & 1 & \\ & & \ddots & \ddots & \ddots \\ & & & 1 & -2 & 1 \\ & & & & \frac{2}{1+r(t)} & \frac{2}{r(t)} \end{bmatrix}, \quad \text{where } r(t) = \frac{(1-t)(N+1)}{t+N+1}$$

¹In the structure of A , there are $L - 1$ blocks T_1 and $M - L$ blocks T_3

\vec{b}_{xx} is a *block* vector of the form

$$\vec{b}_{xx} = \begin{bmatrix} \vec{b}_1(y_1) \\ \vdots \\ \vec{b}_1(y_{L-1}) \\ \vec{b}_2(y_L) \\ \vec{b}_3(y_{L+1}) \\ \vdots \\ \vec{b}_3(y_M) \end{bmatrix}$$

where $\vec{b}_1, \vec{b}_2, \vec{b}_3$ are sparse vectors of sizes $(N+1) \times 1$, $(N+1) \times 1$, and $N \times 1$, respectively

$$\bullet \vec{b}_1(y_j) = \frac{1}{\Delta x_n^2} \begin{bmatrix} u_a(y_j) \\ 0 \\ \vdots \\ 0 \\ u_b(y_j) \end{bmatrix}$$

$$\bullet \vec{b}_3(y_j) = \frac{1}{\Delta x_o^2} \begin{bmatrix} u_a(y_j) \\ 0 \\ \vdots \\ 0 \\ u_b(y_j) \end{bmatrix}$$

$$\bullet \vec{b}_2(y_j) = \frac{1}{h(t)^2} \begin{bmatrix} u_a(y_j) \\ 0 \\ \vdots \\ 0 \\ \frac{2}{r(t)[1+r(t)]} u_b(y_j) \end{bmatrix}, \quad \text{where } r(t) = \frac{(1-t)(N+1)}{t+N+1}$$

A_{yy} is a block tri-diagonal matrix of the form

$$\begin{bmatrix} -2D_1 & D_1 & & & & & & & & & \\ D_1 & -2D_1 & D_1 & & & & & & & & \\ & & \ddots & \ddots & \ddots & & & & & & \\ & & & D_1 & -2D_1 & D_1 & & & & & \\ & & & & D_1 & -2D_1 & B_1 & & & & \\ & & & & & B_2 & -2D_1 & C_1 & & & \\ & & & & & & C_2 & -2D_2 & D_2 & & \\ & & & & & & & D_2 & -2D_2 & D_2 & \\ & & & & & & & & \ddots & \ddots & \ddots \\ & & & & & & & & & D_2 & -2D_2 & D_2 \\ & & & & & & & & & & D_2 & -2D_2 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ \vdots \\ L-2 \\ L-1 \\ L \\ L+1 \\ L+2 \\ \vdots \\ M-1 \\ M \end{matrix}$$

where D_1, D_2 are diagonal square matrices of sizes $N+1$, and N , respectively; B_1, B_2 are bi-diagonal square matrices of sizes $N+1$; and C_1, C_2 are bi-diagonal rectangular matrices of sizes $(N+1) \times N$, and $N \times (N+1)$, respectively

- $D_1 = \frac{1}{\Delta y_o^2} \cdot \mathbf{I}_{N+1}$, where \mathbf{I}_{N+1} is the identity matrix of size $N+1$.

- $D_2 = \frac{1}{\Delta y_o^2} \cdot \mathbf{I}_N$, where \mathbf{I}_N is the identity matrix of size N .

- $B_1 = \frac{1}{\Delta y_o^2} \begin{bmatrix} r_{\bullet} & & & & \\ 1-r_{\bullet} & r_{\bullet} & & & \\ & \ddots & \ddots & & \\ & & 1-r_{\bullet} & r_{\bullet} & \end{bmatrix}$, where $\begin{matrix} r_{\bullet} = 1 - 1 \frac{t}{t+N+1} \\ r_{\bullet} = 1 - 2 \frac{t}{t+N+1} \\ \vdots \\ r_{\bullet} = 1 - (N+1) \frac{t}{t+N+1} \end{matrix}$

$$\begin{aligned}
\bullet \quad B_2 &= \frac{1}{\Delta y_o^2} \begin{bmatrix} 1-r_o & r_o & & & \\ & 1-r_o & r_o & & \\ & & \ddots & \ddots & \\ & & & 1-r_o & r_o \\ & & & & 1-r_o \end{bmatrix}, \quad \text{where} \quad \begin{array}{l} r_o = 1 \frac{t}{N+1} \\ r_o = 2 \frac{t}{N+1} \\ \vdots \\ r_o = N \frac{t}{N+1} \\ r_o = t \end{array}, \\
\bullet \quad C_1 &= \frac{1}{\Delta y_o^2} \begin{bmatrix} r_\bullet & & & & \\ 1-r_\bullet & r_\bullet & & & \\ & \ddots & \ddots & & \\ & & 1-r_\bullet & r_\bullet & \\ & & & 1-r_\bullet \end{bmatrix}, \quad \text{where} \quad \begin{array}{l} r_\bullet = 1 - 1 \frac{1-t}{N+1} \\ r_\bullet = 1 - 2 \frac{1-t}{N+1} \\ \vdots \\ r_\bullet = 1 - (N) \frac{1-t}{N+1} \\ r_\bullet = t + \frac{1-t}{N+1} \end{array}, \\
\bullet \quad C_2 &= \frac{1}{\Delta y_o^2} \begin{bmatrix} 1-r_o & r_o & & & \\ & \ddots & \ddots & & \\ & & 1-r_o & r_o & \end{bmatrix}, \quad \text{where} \quad \begin{array}{l} r_o = 1 \frac{1-t}{t+N+1} \\ \vdots \\ r_o = N \frac{1-t}{t+N+1} \end{array}
\end{aligned}$$

\vec{b}_{yy} is a *block* vector of the form ²

$$\vec{b}_{yy} = \left[\vec{\alpha}^T, \vec{0}, \dots, \vec{0}, \vec{\varphi}^T, \vec{\phi}^T, \vec{0}, \dots, \vec{0}, \vec{\beta}^T \right]^T$$

where $\vec{\alpha}, \vec{\varphi}, \vec{\phi}$ are vectors of sizes $(N + 1) \times 1$, and $\vec{\beta}$ is a vector of size $N \times 1$

$$\bullet \vec{\alpha} = \frac{1}{\Delta y_o^2} \begin{bmatrix} u_c(x_{1_{new}}) \\ u_c(x_{2_{new}}) \\ \vdots \\ u_c(x_{N+1_{new}}) \end{bmatrix}$$

$$\bullet \vec{\beta} = \frac{1}{\Delta y_o^2} \begin{bmatrix} u_d(x_{1_{old}}) \\ u_d(x_{2_{old}}) \\ \vdots \\ u_d(x_{N_{old}}) \end{bmatrix}$$

$$\bullet \vec{\varphi} = \frac{1}{\Delta y_o^2} \begin{bmatrix} (1 - r_{\bullet}) u_a(y_L) \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \text{where} \quad \begin{aligned} r_{\bullet} &= 1 - 1 \frac{t}{t+N+1} \\ r_{\bullet} &= 1 - 2 \frac{t}{t+N+1} \\ &\vdots \\ r_{\bullet} &= 1 - (N + 1) \frac{t}{t+N+1} \end{aligned}$$

$$\bullet \vec{\phi} = \frac{1}{\Delta y_o^2} \begin{bmatrix} (1 - r_{\bullet}) u_a(y_{L+1}) \\ 0 \\ \vdots \\ 0 \\ r_{\bullet} u_b(y_{L+1}) + r_o u_a(y_{L+1}) \end{bmatrix}, \quad \text{where} \quad \begin{aligned} r_{\bullet} &= 1 - 1 \frac{1-t}{N+2} & r_o &= 1 \frac{t}{N+1} \\ r_{\bullet} &= 1 - 2 \frac{1-t}{N+2} & r_o &= 2 \frac{t}{N+1} \\ &\vdots & &\vdots \\ r_{\bullet} &= 1 - N \frac{1-t}{N+2} & r_o &= N \frac{t}{N+1} \\ r_{\bullet} &= 1 - (N + 1) \frac{1-t}{N+2} & r_o &= t \end{aligned}$$

Remark 2.4.1. One also needs to pay some attention for the special cases $t = 1$ and $t = 0$.

A similar calculation was also performed to write the homotopy function when the new point is introduced on a column instead of a row.

²In the structure of \vec{b}_{yy} from above, $\vec{\alpha}^T$ is on the first position, $\vec{\varphi}^T$ is on the $(L - 1)^{th}$ position, $\vec{\phi}^T$ is on L^{th} position, and $\vec{\beta}^T$ is on the M^{th} position.

2.5 *Algorithm for the moving mesh-grid*

During the process of finding all the solutions of a PDE using our homotopy continuation toolbox, we alternate introducing mesh-grid points on rows and columns as we describe in the next algorithm.

Algorithm: The moving mesh-grid for a box-interval (rectangle) in 2-D

- We start with one interior point and solve.
- We introduce a new point on this row and solve. Hence we will get solutions for a mesh with two interior points in a row (they can be seen also as two columns with one interior point each).
- We now introduce a new point in the first column and solve. We will get solutions for a mesh with three interior points: two on the first column and one on the second column.
- We introduce a new point in the second column and solve. We will get solutions for a mesh with four interior points: two on the first column and two on the second column.
- We introduce a new point in the first row now and solve. Hence we will get solutions for a mesh with five interior points: three on the first row and two on the second row.
- The process continues until we have sufficiently many interior points or until the number of solutions we are looking for stabilizes.

Therefore, the procedure of changing the mesh in our homotopy process consists of alternating the following two steps: adding one new interior point to each row at a time until all rows get a new interior point each and adding one new interior point to each column at a time until all columns get a new interior point each.

2.6 Refining a solution

It is clear now that we cannot run our algorithm up to hundreds of interior grid points because the number of solutions (real and complex) we get grows exponentially. We will stop it once we see that the number of real solutions stabilizes. Once we get these real solutions, we need to refine the mesh for them and find the new corresponding solutions. The idea of getting such a new refined solution goes back to interpolation and Newton's method: add interior points on the old mesh-grid and approximate the values of the solution on these points using some interpolation for example; then, using this approximative solution as an initial guess, use Newton's method (again, other numerical methods might be more effective) to find a more *accurate solution* on the this new mesh-grid. We will call it a *refined solution*.

Below is the algorithm for a finding a solution on a new refined mesh-grid, in \mathbb{R}^2 for example, given that our initial domain is a box-interval (rectangle) of the form $\Omega = (a, b) \times (c, d)$.

Algorithm for finding *refined* solutions

1. Start with a solution found using our toolbox on a mesh with $m \times n$ interior grid points (m rows of points, each row having n interior points).
2. Add $(m+1)$ rows of n interior points each and approximate the values of the solution at these new points using linear interpolation (for the first row of new points, use the boundary and the first row of the old points; for the second row of new points, use the first and second rows of old points, etc). Use this solution on $(2m+1) \times n$ interior points as an initial guess for Newton's method and find the *refined solution*.
3. Repeat the previous step, but now add $(n+1)$ columns of $(2m+1)$ interior points each; you will obtain the *refined solution* on an $(2m+1) \times (2n+1)$ interior mesh of grid points.

4. If the new mesh-grid is not sufficiently small, go back to the first step. Otherwise STOP.

Some refined solutions will be presented in the next section.

The reader should also note that this idea of refining a solution is not restricted only to a rectangular domain; it can be easily applied to other kinds of domains. Below is an example of how the number of the interior points increases as we try to refine a solution.

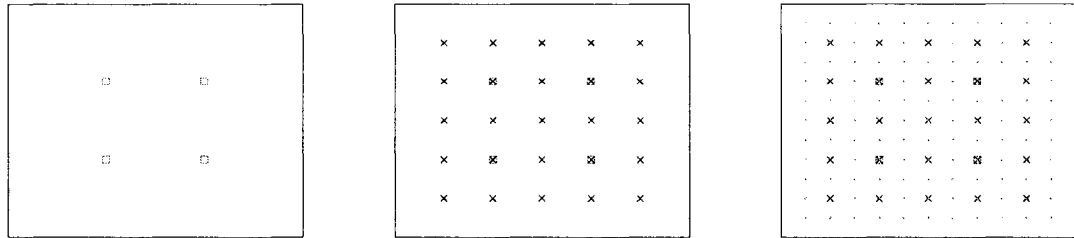


Figure 11: Refining a mesh grid on a box-interval: 2×2 , refined to 5×5 , and then to 11×11 .

2.7 Numerical results for polynomial right hand side

2.7.1 Numerical results for $\Delta u = -(1 + u^2)$

Consider the following 1-D problem

$$\begin{aligned} u'' &= -\lambda(1 + u^2) & \text{on } [0, 1] \\ u(0) &= 0 = u(1) \end{aligned} \tag{31}$$

The bifurcation diagram we obtained using numerical continuation is given in Figure 12.

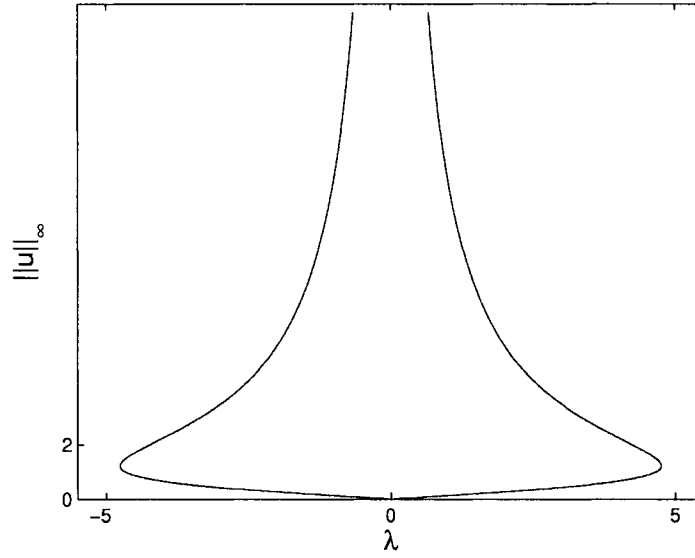


Figure 12: The bifurcation diagram for $u'' = -\lambda(1 + u^2)$ with zero Dirichlet boundary conditions.

The two turning points are reached at $\pm\lambda_*$, where $\lambda_* \approx 4.7547$.

This is just a particular case of the more general boundary value problem

$$u''(t) = -\lambda f(u(t)), \quad 0 \leq t \leq 1, \quad u(0) = 0 = u(1), \tag{32}$$

coming for example, from certain physical problems involving the steady state temperature distribution in a material bounded by two infinite parallel planes, where f is a given function characteristic of the material, and $f(u) \geq 0$ for all $u \geq 0$.

Using our homotopy continuation toolbox for (31), we found that there are exactly two solutions for $|\lambda| < \lambda_*$, a unique solution for $\lambda = \lambda_*$, and no solution for $|\lambda| > \lambda_*$. It is

important to remark that the authors in [1] obtained two real solutions for $|\lambda| < 4$ and no real solution for $|\lambda| > 4$, and hence the wrong idea that $\lambda_* = 4$. Indeed, we did observe that for $\lambda_* - \epsilon < |\lambda| < \lambda_*$ with ϵ small, our homotopy continuation toolbox will give us no real solution for the first few interior points, but as we added more interior points, the two real solutions showed up.

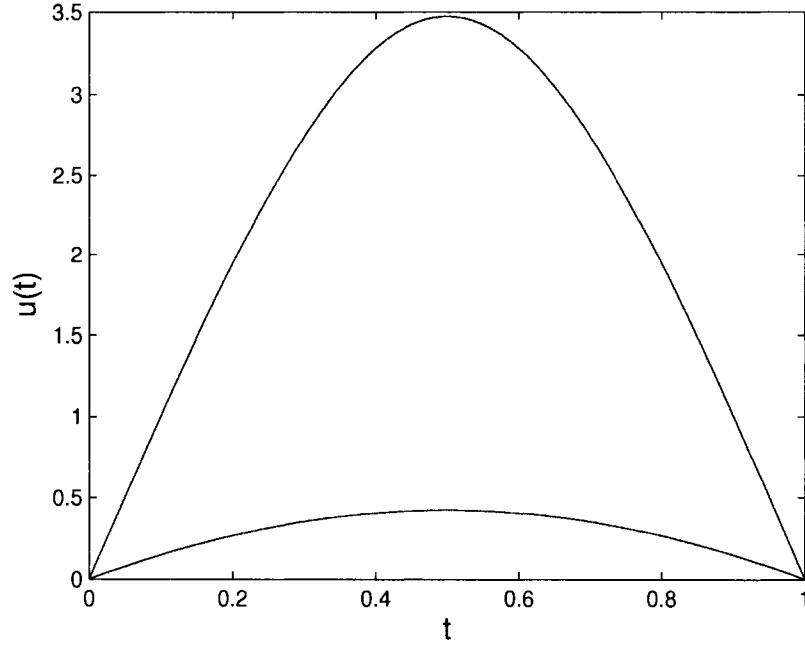


Figure 13: The two solutions with 150 interior grid-points for $u''(t) = -3(1 + u(t)^2)$ with zero boundary conditions.

An important result that might help in constructing a filter for our homotopy numerical continuation can be found in [16]; it states that any nonzero solution of (32) for $\lambda > 0$ is strictly positive and symmetric about the point $t = \frac{1}{2}$. To show the symmetry result, consider the following.

- Any strictly positive solution of (31) has exactly one maximum in $(0, 1)$ (assuming at least two maxima, then $f(u) \geq 0$ for all $u \geq 0$ is contradicted). Let t_0 be the point where $u(t)$ assumes its maximum, i.e. $u(t_0) = \|u\|$, $u'(t) \geq 0$ on $[0, t_0]$, and $u'(t) \leq 0$ on $[t_0, 1]$.

- Let $F(w) = \int_0^w f(v)dv$; then,

multiplying (32) by $u'(t)$ and integrating gives

$$\frac{1}{2}u'(t)^2 = -\lambda F(u(t)) + \lambda F(\|u\|),$$

which can be rewritten as

$$\sqrt{2\lambda} = \frac{|u'(t)|}{\sqrt{F(\|u\|) - F(u(t))}} .$$

Integrating from 0 to t when $0 \leq t \leq t_0$ gives

$$t\sqrt{2\lambda} = \int_0^{u(t)} \frac{dw}{\sqrt{F(\|u\|) - F(w)}}, \quad 0 \leq t \leq t_0 ,$$

and integrating from 0 to t when $t_0 \leq t \leq 1$ gives

$$(1-t)\sqrt{2\lambda} = \int_0^{u(t)} \frac{dw}{\sqrt{F(\|u\|) - F(w)}}, \quad t_0 \leq t \leq 1 .$$

- Now set $t = t_0$ and $u(t) = \|u\|$ to see that $t_0 = \frac{1}{2}$ and $u(t) = u(1-t)$.

Now, let's consider the corresponding problem, but in 2-D.

$$\begin{aligned}\Delta u &= -\lambda(1 + u^2) & \text{on } \Omega = [0, 1]^2 \\ u(0, y) &= u(1, y) = u(x, 0) = u(x, 1) = 0, \forall x, y \in [0, 1]\end{aligned}\tag{33}$$

We first used our homotopy method with continuation in t to track the solutions from $t = 1$ to $t = 0$. We applied the gamma-trick only for solutions that did not converge all the way to $t = 0$ using this continuation in t . For $\lambda = 1$, this method produced one real solution among a total of approximately 2^N solutions, as one can see in the next table where

- n - the number of interior points considered in Ω ;
- L - the row (column) where the new point was introduced;
- row/col - the new point was introduced on a row or a column; if the new point was introduced on a row then

M - the number of rows of points;

N - the number of points in each of the rows $L+1, \dots, M$; hence, rows $1, \dots, L$ each have $N + 1$ points.

Otherwise,

N - the number of columns of points;

M - the number of points in each of the columns $L + 1, \dots, N$; hence, columns $1, \dots, L$ each have $M + 1$ points.

- $SOLS(n)$ - the number of total solutions (real and complex);
- $REAL(n)$ - the number of real solutions;
- $k's$ - the solutions for which we had to apply the *gamma-trick*.

n	L	row/col	M	N	$SOLS(n)$	$REAL(n)$	$k's$
1	1	r	1	1	2	2	
2	1	r	1	2	4	2	
3	1	c	1	2	6	2	5,7
4	2	c	1	2	10	2	
5	1	r	2	2	17	2	17,19
6	2	r	2	2	31	2	3,31,33,34
7	1	c	3	2	62	2	7,61
8	2	c	3	2	121	1	15,16,123
9	3	c	3	2	242	1	29
10	1	r	3	3	484	1	59
11	2	r	3	3	968	1	119
12	3	r	3	3	1936	1	239
13	1	c	4	3	3872	1	479
14	2	c	4	3	7743	1	959
15	3	c	4	3	15485	1	1917
16	4	c	4	3	30966	1	3833

Table 1: The number of solutions for $\Delta u = -(1 + u^2)$ on $\Omega = [0, 1]^2$ with zero Dirichlet boundary conditions. Homotopy with continuation in t was used to track them from $t = 1$ to $t = 0$; if no convergence for a solution, then the gamma-trick was used instead to track it.

We remark that in our research, a solution is considered to be real if the imaginary part at each mesh point is zero to at least eight digits. One can also observe the following:

$$(*) n = (N + 1)L + N(M - L);$$

$$(*) \text{ for } n > 1, \text{ the number of paths tracked at stage } n - 1 \text{ is } d \cdot SOLS(n - 1).$$

During our process, we had to apply the gamma-trick a couple of times. We did not apply the gamma trick to all the solutions because we lose complex solutions, and we end up losing even the real one. From this table, one can also see that we need to have some filters such that we can throw away *spurious* solutions as the number of interior grid points increases. We observed that out of those 30966 solutions for 16 interior points, there are exactly one real and four complex ones which are invariant to the dihedral group of symmetries (flip with respect to $x = \frac{1}{2}$, rotation with $\frac{\pi}{2}$). Below we have plotted these five solutions (the real part of them) and a sixth one (random out of the remaining 30961), as we refined them. It is also remarkable that, as we refined the grid, we observed that only

these five solutions did not modify their max. One can easily see that the sixth one, for example, had a big change in the max as we doubled the number of interior grid-points on each axis. This can be used in our toolbox to build a filter.

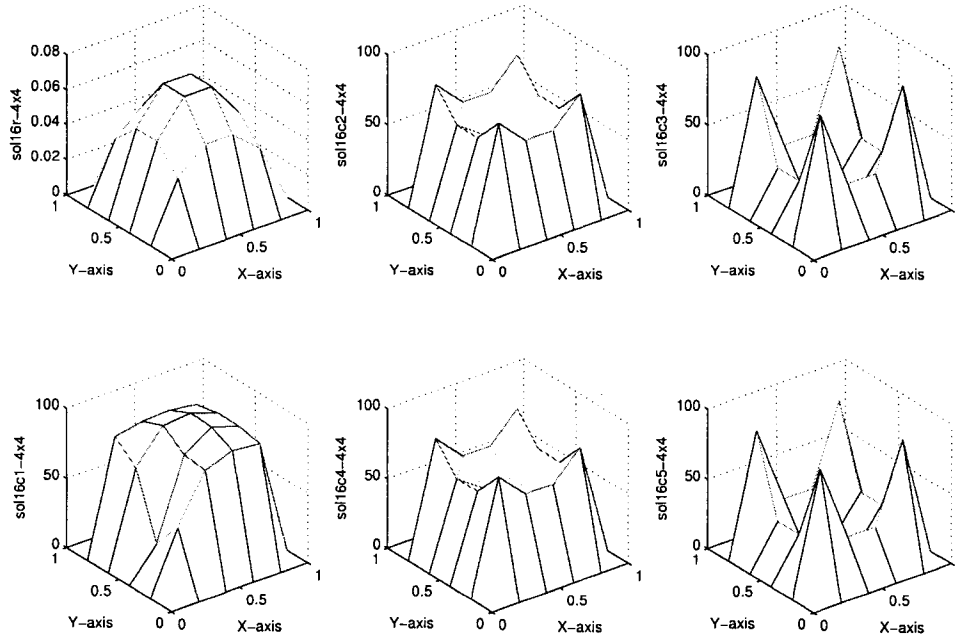


Figure 14: Six solutions for $\Delta u = -(1 + u^2)$ with 4×4 interior points.

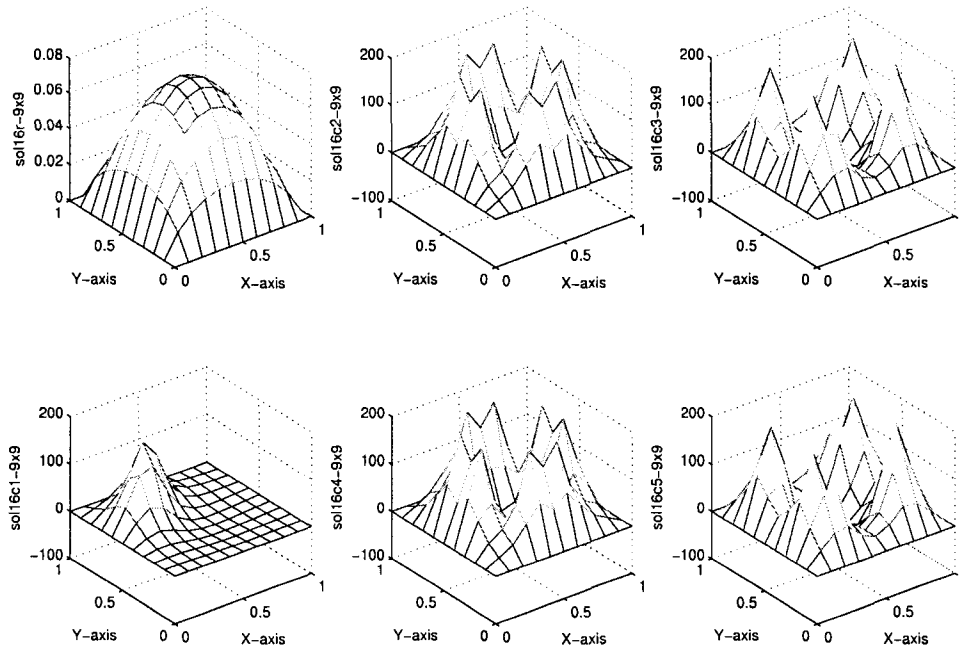


Figure 15: The six solutions for $\Delta u = -(1 + u^2)$ with 4×4 interior points refined to 9×9 .

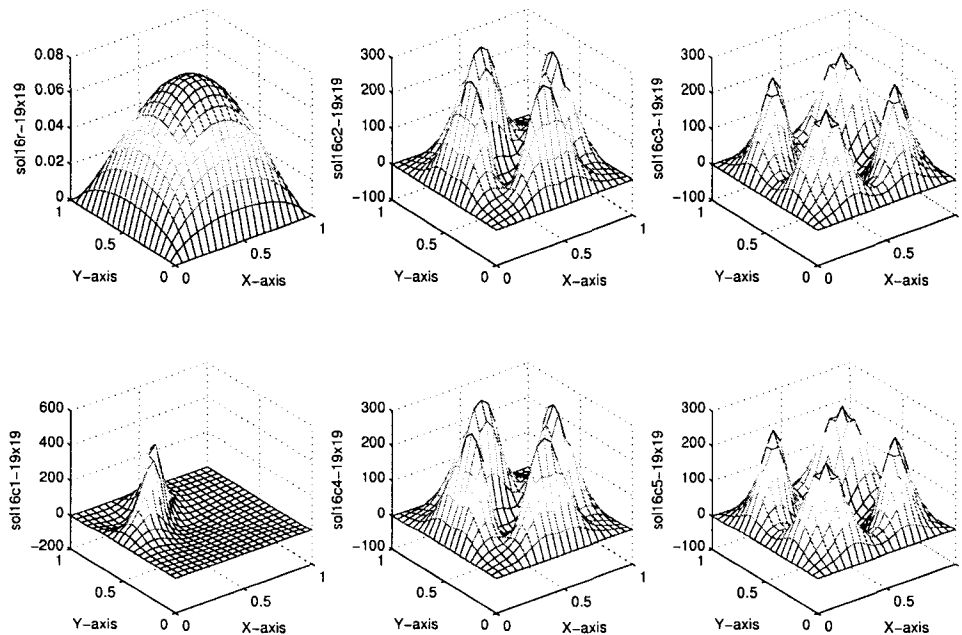


Figure 16: The six solutions for $\Delta u = -(1 + u^2)$ with 4×4 interior points refined to 19×19 .

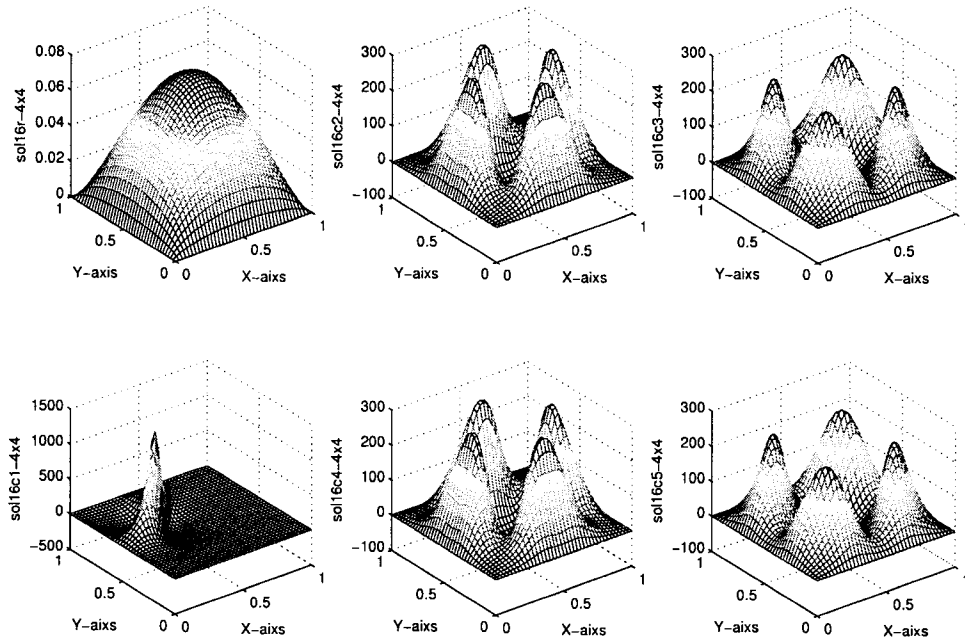


Figure 17: The six solutions for $\Delta u = -(1 + u^2)$ with 4×4 interior points refined to 39×39 .

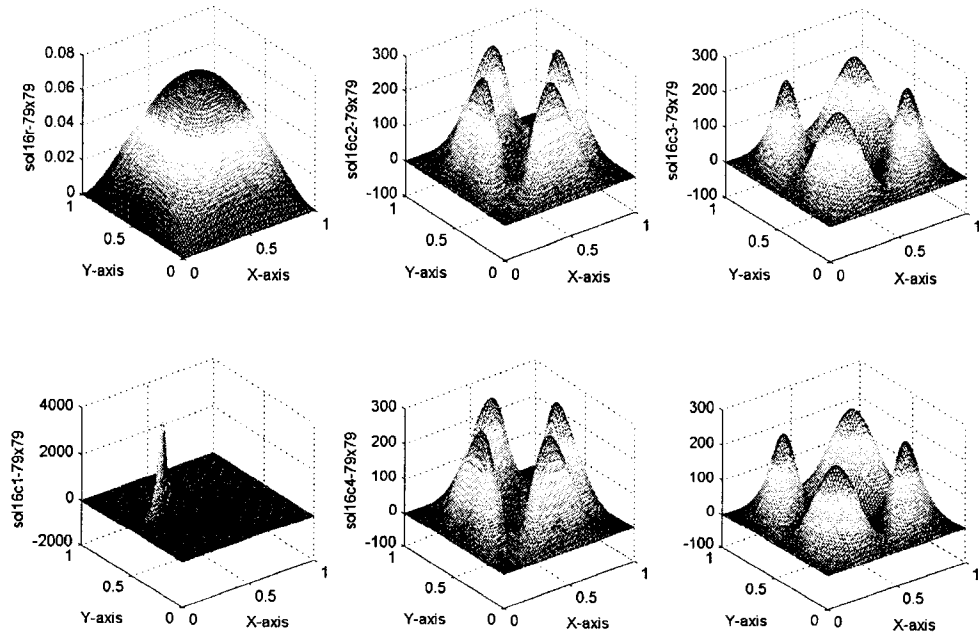


Figure 18: The six solutions for $\Delta u = -(1 + u^2)$ with 4×4 interior points refined to 79×79 .

We refined the real solution that we have gotten to 39×39 interior points and then, starting with this new refined one, we decided to apply arclength continuation to see its evolution as λ in (33) modifies. The result was not very surprising as one can see in the bifurcation diagram given in Figure 19.

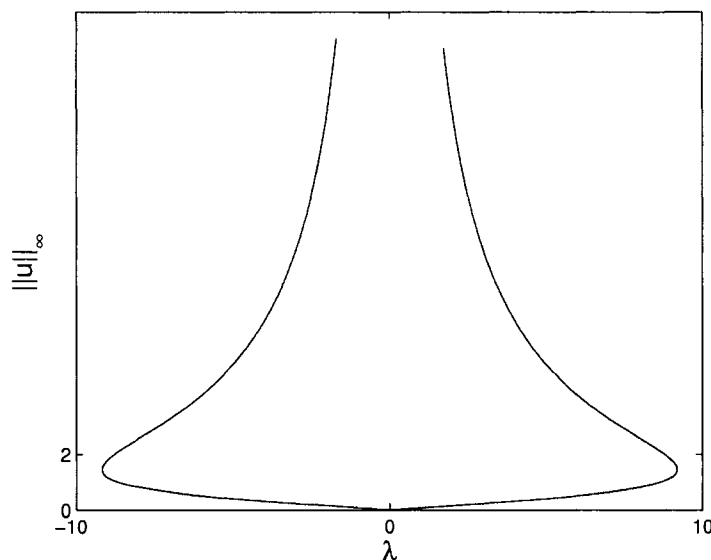


Figure 19: The bifurcation diagram for $\Delta u = -\lambda(1 + u^2)$ with zero Dirichlet boundary conditions.

The two turning points in this diagram are attained at $\pm\lambda_*$, with $\lambda_* = 9.1890$.

As one can see, we should have obtained two real solutions as in the 1-D case. What happened? Tracking the solutions of the homotopy using continuation in t , combined with the gamma-trick when necessary did not handle the turning points well and gave us sometimes identical solutions at $t = 0$ even if the starting solutions at $t = 1$ were different. For example, after finishing introducing two points, we end up with 4 different solutions for our homotopy H_2 at $t = 0$. Now, we start introducing a third interior point and therefore, using a similar equation with (8) but corresponding to our 2-D problem, these previous 4 solutions will give birth to 8 different new starting solutions for our homotopy H_3 at $t = 1$. These 8 solutions have to be tracked all the way down to $t = 0$. We noticed that 6 of them were nicely tracked to $t = 0$ using continuation in t , but 2 of them (the 5th and the 7th)

did not reach $t = 0$ (the step size in t got very small near $t = 0.1003$). We took these 2 solutions again from $t = 1$ and track them but using this time the gamma-trick. The result was satisfactory: these 2 solutions could be tracked this time all the way down to $t = 0$, but here, they were the same with 2 other solutions from the 6 we got before. So, we found 6 solutions for 3 interior points. We have continued this process until we reached 16 interior points and the results were presented in Table 1.

We have also observed that if we were to use the gamma-trick in tracking all the solutions, we end up losing even the real one that we got for 16 interior points. For this reason we decided to introduce the arclength continuation in tracking the solutions as t goes from 1 to 0 and handle the turning points as we described in the section 1.3. With this method of tracking, the results improved a lot as we can see in the next table. This time for example, tracking the starting 5th solution of H_3 for $n = 3$ interior points, we were able to go around the turning point at $t = 0.1003$ and not surprisingly arrive back at the starting 7th solution of H_3 at $t = 1$ (we stayed on the same branch using arclength continuation). Using now the theory presented in the section 1.3, we were able to switch branches at the turning point and finally arrive at $t = 0$ where we obtained a solution which was not anymore identical with anyone of the other 6 as in the continuation in t tracking case. After tracking all the solutions for $n = 3$ interior points, we got a total of 8 solutions which is the maximum possible number of solutions that we can get (by Bezout's theorem).

Using a tracking method based on arclength continuation [14, 17] combined with the 'end game' presented in section 1.4 we obtained the results presented in Table 2. When tracking a path, if we did not reach the goal $t = 0$ because of a turning point, we proceeded to switch branches using the tangents motivated by the Theorem 1.3.8 and the Corollary 1.3.9.

n	L	r/c	M	N	$SOLS$ (n)	$REAL$ (n)	Sols. with same turning points	The value of t at the turning points
1	1	r	1	1	2	2		
2	1	r	1	2	4	2		
3	1	c	1	2	8	2	(5,7)	0.1003
4	2	c	1	2	16	2	(11,15)	0.0036
5	1	r	2	2	32	6	(18,26) (2,10) (23,31)	0.9245 0.8573 0.3553
6	2	r	2	2	64	4	(47,51) (48,52) (3,19) (35,63)	0.9831 0.9438 0.9242 0.3592
7	1	c	3	2	128	4	(7,39) (8,72) (71,127)	0.9739 0.8996 & 0.0175 0.3853
8	2	c	3	2	256	4	(79,143) (80,144) (15,255)	0.9970 0.9616 0.4410
9	3	c	3	2	512	2	(31,511)	0.3820
10	1	r	3	3	1024	2	(63,1023)	0.4828
11	2	r	3	3	2048	2	(127,2047)	0.5638
12	3	r	3	3	4096	2	(255,4095)	0.4858
13	1	c	4	3	8192	2	(511,8191)	0.4945
14	2	c	4	3	16384	2	(1023,16383)	0.5907
15	3	c	4	3	32768	2	(2047,32767)	0.5920
16	4	c	4	3	65536	2	(4095,65535)	0.4985

Table 2: The number of solutions for $\Delta u = -(1 + u^2)$ on $\Omega = [0, 1]^2$ with zero Dirichlet boundary conditions. Arclength continuation was used to track all the solutions of the homotopy associated to the discretization of this problem.

For this table, the following observation can be made.

- For each $n = 1, \dots, 16$ interior points, we have obtained exactly the maximum possible number of solutions that the homotopy can have (by Bezout's theorem).
- Out of these, the number of real solutions has been stabilized to 2 as n increased.
- The behavior of the 2^n tracked paths has been also stabilized: every time we have added a new point, the two out of four real solutions at $t = 1$ (the $(2^n - 1)^{th}$ and the

$(2^{n-4} - 1)^{th}$ ones) will be on the same bifurcation branch that will not reach $t = 0$. Using the theory from section 1.3, we are able to switch the branches and reach the goal $t = 0$ where these two solutions were complex. The $(2^n)^{th}$ and $(2^{n-4})^{th}$ solutions were the only real solutions for which their paths behaved well all the way down to $t = 0$ (they remained real as we tracked them from $t = 1$ to $t = 0$).

Having also an idea about the bifurcation diagram presented before (Figure 19), we conclude that the problem (33) has indeed only 2 real solutions.

Remark 2.7.1.1. Using arclength continuation in a smart way (for example, not letting the angle between two consecutive tangents get bigger than a specific value) we were able to handle well turning points that were very close to our initial or goal values of t ($t = 1$, and $t = 0$ respectively): see for example the turning points at $t = 0.0036$ and $t = 0.9970$ for $n = 4$ and $n = 8$ interior points, respectively. In the past, we would have not even seen that there was a turning point at $t = 0.0036$ for $n = 4$ interior points. This is because we would have used the first idea of ‘end game’ of tracking presented in section 1.4 combined with continuation in t instead of arclength continuation and therefore, on the last step of tracking done to reach exactly $t = 0$, we would have jumped unintentionally on a different branch which was not connected in any way to the one we were tracking; this is the way we were obtaining at $t = 0$ same solutions from different initial ones. For the other case (turning points near our initial $t = 1$), we would have jumped unintentionally from the beginning on a different branch than the one we were supposed to follow, because our initial step would have been too big.

Remark 2.7.1.2. In this example, as in fact in all the ones we studied, during our tracking we met all the possible paths presented in section 1.4, except the third type.

Remark 2.7.1.3. Out of these 65536 solutions found on a mesh with 16 interior points (see Table 2), only the two real ones and four other complex ones (the ones met also in figures 14, 15, 16, 17, and 18) did not suffer big modifications as we refined the grid. One of these

two real solutions is plotted in all these five figures, and the other one looks similar, but has a higher peak.

2.7.2 Numerical results for $\Delta u = -\lambda(1 + u^2)$, $\lambda \in \{9, 10\}$

We also considered the equation (27) for $f(u) = -\lambda(1 + u^2)$ on $\Omega = [0, 1]^2$ with zero Dirichlet boundary conditions and $\lambda \in \{9, 10\}$. For $\lambda = 10$, our toolbox produced no real solutions out of 2^N complex ones. For $\lambda = 9$ (a value close to the turning point $\lambda_* = 9.1890$ from the previous bifurcation diagram), we got no real solutions for the first 4 interior points; one real solution appeared right after introducing the 5th interior point. The second one appeared much later.

This is what was expected of course: as we take values for λ close to the turning point $\lambda_* = 9.1890$, the real solutions will appear after introducing more interior points.

2.7.3 The need for complex solutions?

Consider again the equation (27) for $f(u) = -\lambda(1 + u^2)$ on $\Omega = [0, 1]^2$ with zero boundary conditions and $\lambda = -1$. Examining the Table 2 from 2.7.1, the question as how can we find the complex solutions of this problem arises? It is clear that our toolbox gives also lots of complex solutions, but we have to be careful: lots of them do not have correspondents in the continuous case (they are just spurious solutions).

It looks like not much has been done in this sense in the literature. There are some results, but only about the real solutions, including the following famous Gidas, Ni, Nirenberg Theorem (see also [18]).

Theorem 2.7.3.1 (Gidas, Ni, Nirenberg). *Every real solution of*

$$\Delta u = f(u), \quad u > 0 \text{ in } \Omega, \quad u = 0 \text{ on } \partial\Omega \quad (34)$$

is radially symmetric provided that $\Omega = B_1(0) \subset \mathbb{R}^n$ is a ball and $f : [0, \infty] \rightarrow \mathbb{R}$ is locally Lipschitz continuous. If $B_1(0)$ is replaced by an n -dimensional rectangle

$$\Omega = [-a_1, a_1] \times \dots \times [-a_n, a_n]$$

then, an analogous symmetry result (Berestycki and Nirenberg) holds. It states that

$$u(x_1, \dots, x_{j-1}, -x_j, x_{j+1}, \dots, x_n) = u(x_1, \dots, x_{j-1}, x_j, x_{j+1}, \dots, x_n), \quad \forall j = 1, \dots, n.$$

There are no results about the complex solutions for such systems. One way to find these complex solutions is to apply our homotopy continuation and then eliminate maybe the spurious ones. Another way is to try to solve the following *equivalent system*

$$\begin{aligned} \Delta u_{\Re} &= 1 + u_{\Re}^2 - u_{\Im}^2 \\ \Delta u_{\Im} &= 2u_{\Re}u_{\Im} \quad \text{on } \Omega \subset \mathbb{R}^2 \\ u_{\Re}|_{\partial\Omega} &= 0 = u_{\Im}|_{\partial\Omega} \end{aligned} \tag{35}$$

where u_{\Re}, u_{\Im} are real solutions.

One can easily observe that this system (35) and the one considered in subsection 2.7.1 are indeed equivalent. But the drawback of this system is the fact that it is a coupled one and our toolbox needs to be modified to solve something like this.

CHAPTER III

SOLUTION OF AN OPEN CONJECTURE

In this chapter, using the toolbox described in the previous chapter, we will give a solution to McKenna's open problem.

In 2003, Breuer, McKenna and Plum published a paper [5] in which they discovered the following theorem for which they gave a computational multiplicity proof.

Theorem 3.0.1. *The equation*

$$\Delta u + u^2 = 800 \sin \pi x \sin \pi y \quad \text{in } \Omega, \quad u = 0 \quad \text{on } \partial\Omega \quad (36)$$

where $\Omega = (0, 1) \times (0, 1)$, has at least four solutions.

Our toolbox is built such that it will give all the solutions of (25) not only for $f(u)$ being a polynomial of u , but also for any function $f(x, y, u)$ that is a polynomial as a function of u , because even in this case, the corresponding equation from (9) for our 2-D BVP is still a polynomial equation in one unknown. For this particular problem, we found using our homotopy continuation that there are exactly four essentially real different solutions. Using the toolbox we also got real solutions that are rotations or reflections of these, and therefore, after some filtering, we got only the four truly distinct real solutions that appear in Figure 20.

The two solutions from the left column are fully symmetric (i.e. symmetric with respect to reflections about the axes $x = \frac{1}{2}$, $y = \frac{1}{2}$, $x = y$, and $x = 1 - y$), the solution from the upper right corner is only symmetric with respect to reflection about $y = \frac{1}{2}$, and the solution from the lower right corner is only symmetric with respect to reflection about $x + y = 0$.

The peaks of the positive and negative fully symmetric solutions have the values 61.776, and -21.358 respectively; the peak of the solution that is symmetric with respect to the

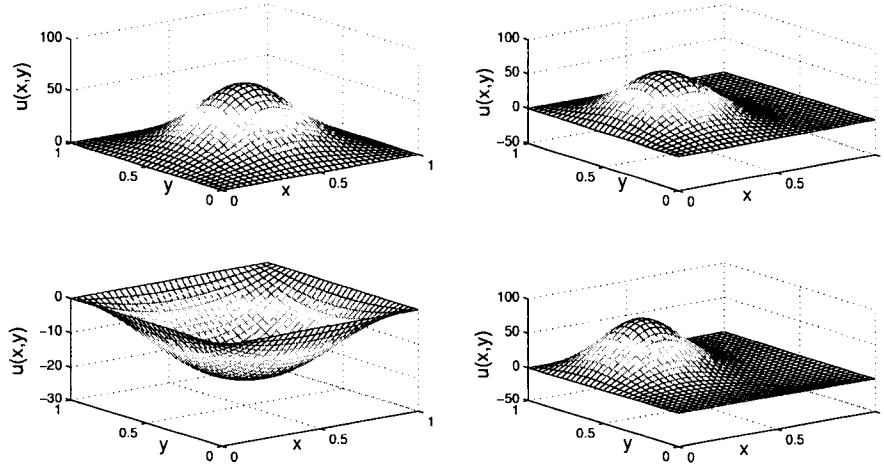


Figure 20: The four real solutions for $\Delta u + u^2 = 800 \sin \pi x \sin \pi y$.

reflection about $y = \frac{1}{2}$ has the value 69.923 and is attained at $(x, y) = (\frac{5}{16}, \frac{1}{2})$ (the 20×31 mesh point); the peak of the solution that is symmetric with respect to the reflection about $x + y = 0$ has the value 76.321 and is attained at $(x, y) \approx (\frac{1}{3}, \frac{2}{3})$ (somewhere between the $(21 - 22) \times (42 - 43)$ mesh points) (all these values were found out using a mesh with 63×63 interior points).

But of course, this particular PDE can be generalized to

$$\Delta u + u^2 = \lambda \sin \pi x \sin \pi y \quad \text{in } \Omega, \quad u = 0 \quad \text{on } \partial\Omega. \quad (37)$$

Starting with those 4 solutions found for $\lambda = 800$ and refined to mesh-grid with 31×31 interior points, we used numerical continuation and constructed the bifurcation diagram associated with this problem (see Figure 21).

We found a turning point T at $\lambda_T \approx -133.3$ (one eigenvalue of the Jacobian changes sign) and a symmetry breaking bifurcation S at $\lambda_S \approx 587.7$ (a pair of two more eigenvalues of the Jacobian (a double eigenvalue) is changing sign).

In order to test our findings, we performed homotopy continuation for a few different values of λ and indeed, we obtained that there are no real solutions for $\lambda < \lambda_T$, two real solutions if $\lambda_T < \lambda < \lambda_S$, and four real solutions if $\lambda > \lambda_S$.

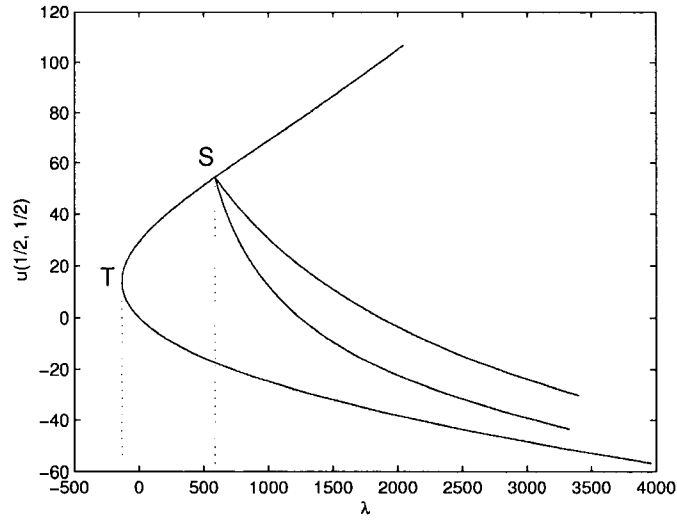


Figure 21: The bifurcation diagram for $\Delta u + u^2 = \lambda \sin \pi x \sin \pi y$ with Dirichlet boundary conditions.

We went with values of λ up to 8000 and we also looked at the eigenvalues of the Jacobian. As expected, one eigenvalue passed through 0 at the turning point, 2 more at the bifurcation point and there is no indication that another eigenvalue will approach zero as λ increases.

We also remark that Breuer, McKenna and Plum state on page 267 in [5] that *the stronger version of the conjecture suggests that as $\lambda \rightarrow \infty$, more solutions are created as bifurcations from the positive curve, considerably further up the positive branch* (the weaker version says that there are at least four solutions). As they, we also did not detect this phenomenon numerically.

We conclude this Chapter by mentioning that the Mountain Pass Algorithm used by McKenna, Breuer, and Plum [5] to find the four solutions of (36) is in general restricted to semilinear partial differential equations whose solutions are the critical points of an associated functional which must satisfy the hypothesis of the Mountain Pass Theorem (see appendix B for more details). Our approach works for a much larger group of problems and does not require any starting solution.

CHAPTER IV

HOMOTOPY CONTINUATION FOR NONPOLYNOMIAL NONLINEARITY

The problem of generalizing our toolbox to nonpolynomial nonlinearities in our partial differential problems arises naturally at this stage. In all the previous cases (polynomial nonlinearity), by using companion matrices we were able to find all the solutions of the polynomial equation that resulted from the introduction of a new mesh point. Now, we have to deal with a new issue, namely finding all the solutions (real and complex) for a nonpolynomial equation. Since there may be infinitely many complex solutions, it is necessary to seek solutions in a bounded region, which is taken to be a compact rectangular $(2d)$ -cell, where d represents the dimension of the equation. Therefore, we have to introduce in this chapter the notion of cellular exclusion methods, with the help of which, one is able to find all real solutions to systems of equations within a rectangular n -cell, provided the nonlinearity satisfies some very general conditions. This approach is applied in this chapter to the familiar Bratu equation in one dimension.

A second issue that arises is that one does not any longer know how many solutions the discretized system has, or indeed, if there are any, or whether there are only finitely many solutions. But, if we restrict ourselves to a bounded n -cell, this problem disappears. This is in general the generic situation for most of the problems arising from real life applications for which solutions of the variables need to be sought in a specific bounded cell.

4.1 Bratu Problem and the Necessity of Exclusion Tests

Recall that the toolbox developed by Allgower, Sommease, Bates and Wampler in [1] was able to find all the solutions of a 1-D boundary value problem of the form

$$\begin{aligned} u'' &= f(x, u, u') \quad \text{where } x \in [a, b] \\ u(a) &= \alpha, u(b) = \beta \end{aligned}$$

where f is a polynomial of u . We now consider the case in which f is not a polynomial of u . As an example, we will look at the 1-D Bratu Problem:

$$\begin{aligned} u_{xx} + \lambda \exp(u) &= 0 \quad \text{on } [0,1] \\ u(0) &= u(1) = 0. \end{aligned} \tag{38}$$

Associate the homotopy function from (6) and (7) to this problem. To be able to use our algorithm for finding all the solutions of (38) for a specific value of λ , we need to be able to find all the roots of the equation (8) which, for this case, can be rewritten as

$$u_N - 2u_{N+1} + \lambda \left(\frac{1}{N+1} \right)^2 \exp(u_{N+1}) = 0. \tag{39}$$

Therefore now, we have to concentrate on finding all the real solutions for the equation

$$\exp(u) - au + b = 0, \tag{40}$$

where a and b are positive constants, $a = 2(N+1)^2/\lambda$, $b = u_N(N+1)^2/\lambda$. After a little bit of calculus, one can easily see that the number of real zeros for such an equation is either 2, 1 or 0, depending upon whether $a * (1 - \ln(a)) + b$ is negative, zero or positive, respectively (see Figure 22). One can easily implement a method to find the zeros of this equation and integrate that into our homotopy continuation toolbox (it might help to see that $\frac{b}{a}$ is a lower bound for these zeros, if they exist).

Using our algorithm for this problem with an black box for solving (40), we were very happy to see that we obtained only 2 real solutions every time we introduced a new grid point, and therefore we could even refine our solutions up to hundred or thousands of interior grid-points using this method.

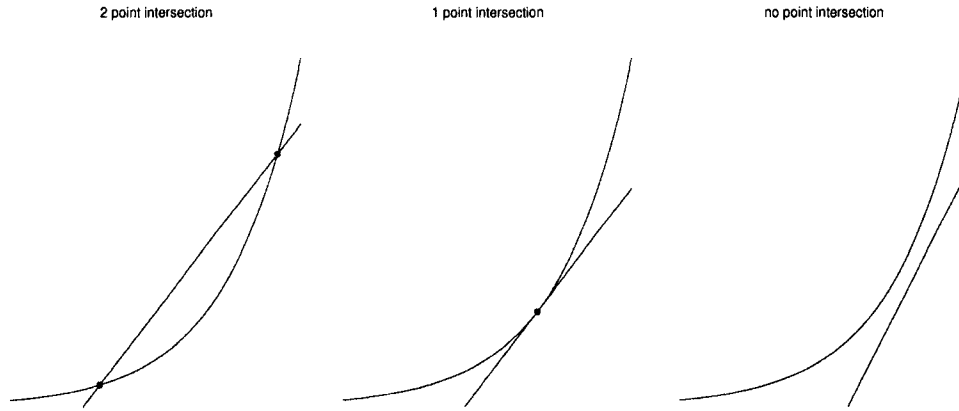


Figure 22: The number of real solution for (40).

Keep in mind that we are solving now (40) only for real solutions, and therefore, every time we introduce a new grid point one expects to double the number of solutions from the previous step. We did obtain that behavior when we were using our toolbox for polynomial case of degree two and looked also for complex solutions. Our method worked very well for $\lambda \in [0, 2.1]$ and it is well known that the bifurcation diagram for the Bratu Problem looks like the one presented in Figure 23, where there is a turning point at $\lambda_* \approx 3.5127$.

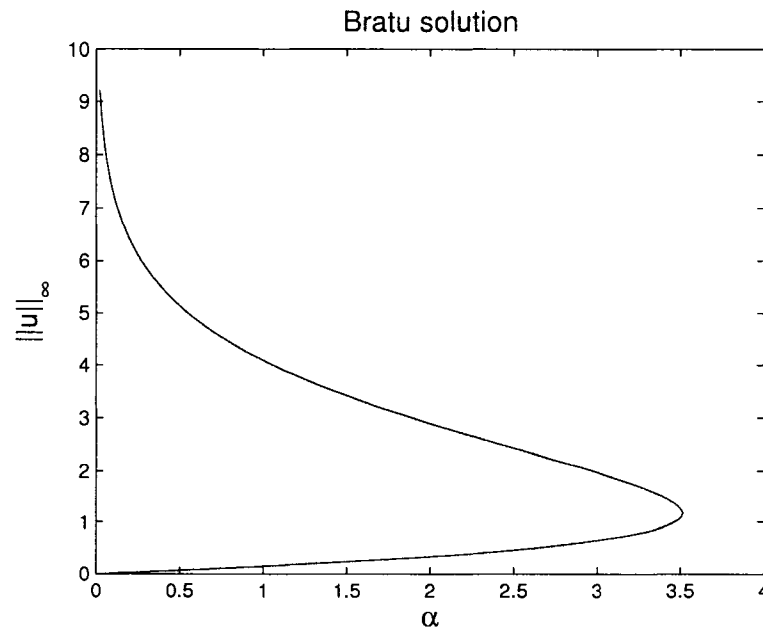


Figure 23: The bifurcation diagram for the Bratu Problem.

In the interval $[2.1, \lambda_*]$ the difficulty we encountered was the fact that the equation (40) did not have any *real* zeros for small values of N , and therefore we could not start using our homotopy. This reminds us of a fact encountered in the polynomial case, that sometimes real solutions will be born from complex ones as we increase the number of interior points. Therefore, it is not sufficient to know how to solve (40) only for real solutions, but also for the complex ones.

Writing $u = u_{\Re} + iu_{\Im}$ and $b = b_{\Re} + ib_{\Im}$, (40) becomes

$$\begin{aligned}\exp(u_{\Re}) \cos(u_{\Im}) - au_{\Re} + b_{\Re} &= 0 \\ \exp(u_{\Re}) \sin(u_{\Im}) - au_{\Im} + b_{\Im} &= 0.\end{aligned}$$

The necessity of solving this system for the reals u_{\Re} and u_{\Im} led us to consider the exclusion algorithms. Using these algorithms, we were able to find all the solutions u_{\Re} and u_{\Im} of this system in a box-interval.

4.2 Exclusion Tests

In this section we will give some background about the exclusion algorithms that we will use in the thesis (see also [10]). The exclusion tests are well known as a very useful tool for finding all the solutions of a nonlinear system of equations over a compact domain. They are also used sometimes for finding the global minima of a function. The ideas go back to Moore [19], 1977, but the real research into this direction began sometimes in the middle of the '90s when few researchers from China developed some cell exclusion algorithms to find all the solutions of a nonlinear system [20, 21, 22, 23]. Then Georg and some of his collaborators introduced and analyzed some new tests for finding the zeros and the global minima of a function over a compact domain in which the efficiency was improved a lot [24, 4, 10]. A min-max test for the exclusion algorithm was recently developed by Syam in [25].

4.2.1 Introduction to Exclusion Algorithms

In \mathbb{R}^n and $\mathbb{R}^{m \times n}$ we use the component-wise “ \leq ” as a partial ordering, “ $|\cdot|$ ” as the component-wise absolute value, and “ $\|\cdot\|_\infty$ ” as the max norm. For example, for two matrices $A, B \in \mathbb{R}^{m \times n}$, the symbol $A \leq B$ means that $A(i, j) \leq B(i, j)$ for $i = 1, \dots, m$, $j = 1, \dots, n$.

Definition 4.2.1.1. An interval σ in \mathbb{R}^n is a rectangular box of the form

$$\sigma = [m_\sigma - r_\sigma, m_\sigma + r_\sigma] = \{x \in \mathbb{R}^n : m_\sigma - r_\sigma \leq x \leq m_\sigma + r_\sigma\},$$

where $m_\sigma, r_\sigma \in \mathbb{R}^n$ are called the **midpoint** and the **radius** of σ respectively, with $r_\sigma(i) > 0, \forall i = 1, \dots, n$. Also, $m_\sigma - r_\sigma$ and $m_\sigma + r_\sigma$ are called the **lower** and **upper corners** respectively.

Definition 4.2.1.2. Let $\sigma \in \mathbb{R}^n$ be an interval and $F : \sigma \rightarrow \mathbb{R}^n$ be a function defined on σ . A test

$$T_F(\sigma) \in \{0, 1\} \quad \text{where} \quad 0 \equiv \text{no} \quad \text{and} \quad 1 \equiv \text{yes}$$

is called an **exclusion test** for F on σ iff $T_F(\sigma) = 0$ implies that F has no zero point in σ .

Therefore, $T_F(\sigma) = 1$ is a **necessary** condition for F to have a zero point in σ .

If an exclusion test is given (assuming that $T_F(\sigma)$ is available for any subinterval σ of some initial interval Λ on which F is defined), then we can recursively bisect intervals and discard the ones which yield to a negative test. This is the basic idea of an **Exclusion Algorithm**.

Remark 4.2.1.3. In order to simplify and unify our efficiency investigations, we will consider only the strategy of **cyclic bisections** of the intervals along subsequent axes. We say that *we have reached a new bisection level* whenever one cycle of bisections is accomplished. We also think of an exclusion algorithm as performing a fixed number of bisection levels. We will denote by Γ_l the list of **the intervals generated by the algorithm on the l -th level**, which are in fact the intervals that have not been discarded after l bisection levels.

Remark 4.2.1.4. It is obvious that if $\Gamma_l = \emptyset$ for some level l , then the algorithm has shown that there are no zero points of F in the initial interval Λ .

Exclusion Algorithm

```

 $\Gamma \leftarrow \{\Lambda\}$ 
for  $l = 1 : \text{maximal\_level}$ 
  for  $a = 1 : n$ 
    let  $\tilde{\Gamma}$  be obtained by bisecting each  $\sigma \in \Gamma$  along the axis  $a$ 
    for  $\sigma \in \tilde{\Gamma}$ 
      if  $T_F(\sigma) = 0$ 
        drop  $\sigma$  from  $\tilde{\Gamma}$  ( $\sigma$  is excluded)
     $\Gamma \leftarrow \tilde{\Gamma}$ 
   $\Gamma_l \leftarrow \Gamma$ .
```

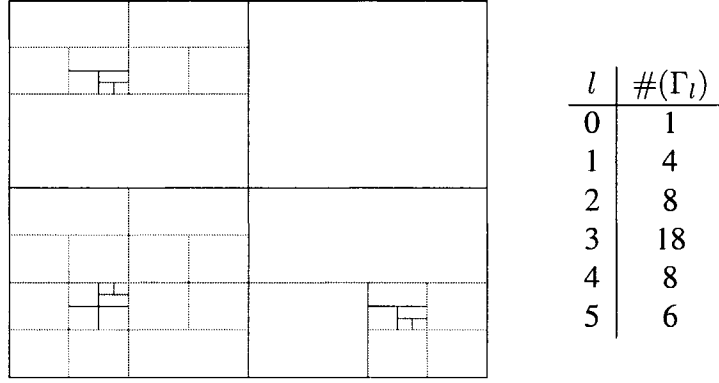


Figure 24: Illustration of Bisection Levels for a box-interval $\sigma \subset \mathbb{R}^2$

The exclusion tests we discuss are applied component-wise on vector-valued function $F : \sigma \rightarrow \mathbb{R}^n$. Therefore, we only need to consider an exclusion test for a scalar-valued function $f : \sigma \rightarrow \mathbb{R}$ and then combine such (possibly different types of) exclusion tests to obtain an exclusion test for a vector-valued function $F = \{f_i\}_{i=1:n} : \sigma \rightarrow \mathbb{R}^n$ by setting

$$T_F(\sigma) := \prod_{i=1}^n T_{f_i}(\sigma) \quad .$$

Hence, we can concentrate our attention on scalar functions $f : \sigma \rightarrow \mathbb{R}$ when designing exclusion tests. We also need good exclusion tests which are computationally inexpensive but relatively tight, because otherwise too many intervals remain undiscarded on each bisection level and this will lead to significant numerical inefficiency. Two simple exclusion tests were given in [21].

Exclusion Test 1

Let $L > 0$ be a Lipschitz constant for f on the interval σ . Then

$$f(m_\sigma) \leq L \|r_\sigma\| \tag{41}$$

is an exclusion test for f on σ .

Exclusion Test 2

If $f = g - h$ is the difference of two increasing functions on σ , then

$$h(m_\sigma - r_\sigma) \leq g(m_\sigma + r_\sigma) \quad \text{and} \quad h(m_\sigma + r_\sigma) \geq g(m_\sigma - r_\sigma) \quad (42)$$

is an exclusion test for f on σ .

Another exclusion test using power series was given in [22]. To write this, we first need some notation.

Notations 4.2.1.5. Let \mathbb{Z}_+ be the set of the nonnegative integers. For a multi-index $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{Z}_+^n$ we will consider the following notations.

- The length of α defined by $|\alpha| := \sum_i \alpha_i$.
- The factorial of α defined by $\alpha! := \prod_i \alpha_i!$.
- If $x \in \mathbb{R}^n$, then we define $x^\alpha := \prod_i x_i^{\alpha_i}$.
- Partial derivatives $\partial^\alpha := \frac{1}{\alpha!} \prod_i \partial_i^{\alpha_i}$.

On the interval $[0, 1]$, we also introduce the probability measures

$$\omega_k(dt) := k(1-t)^{k-1}dt.$$

Definition 4.2.1.6. For two power series $f(x) = \sum_\alpha f_\alpha x^\alpha$ and $g(x) = \sum_\alpha g_\alpha x^\alpha$ we define

$$f \prec\prec g \iff |f_\alpha| \leq g_\alpha \text{ for all } \alpha.$$

Having this definition, the exclusion test in [22] can be formulated as below.

Exclusion Test 3

If $f \prec\prec g$ and if the power series for g converges on σ , then

$$|f(m_\sigma)| \leq g(|m_\sigma| + r_\sigma) - g(|m_\sigma|) \quad (43)$$

is an exclusion test for f on σ .

For all these tests, the following complexity result was also shown in [21], [22].

Theorem 4.2.1.7. *Let $\Lambda \subset \mathbb{R}^n$ be an interval, $F : \Lambda \rightarrow \mathbb{R}^n$ sufficiently smooth and zero a regular value of F . Then there is a constant $C > 0$ such that the exclusion algorithm started in Λ generates no more than C intervals on each bisection level, i.e. $\#(\Gamma_l) \leq C$ independent of l .*

Remark 4.2.1.8. The constant C can be very big and some numerical experiments have shown that the exclusion algorithms in (41), (42) and (43) are not tight enough for more demanding non-linear systems, such as those occurring typically in engineering.

The test we used in our research is presented in [10]. We can remark that even higher singularities in a solution point (as long as the solution point is isolated) does not destroy the complexity addressed in the previous theorem if we use these improved exclusion tests.

4.2.2 Dominant Functions

Using the same notation as introduced in the previous section, we can easily write the Taylor's formula with $k > 0$ and integral reminder for a function f :

$$f(m+h) = f(m) + \sum_{0 < |\alpha| < k} \partial^\alpha f(m) h^\alpha + \sum_{|\beta|=k} \int_0^1 \partial^\beta f(m+th) \omega_k(dt) h^\beta \quad (44)$$

Definition 4.2.2.1. *Let $\sigma \subset \mathbb{R}^n$ be an interval. We denote:*

$$\mathcal{A}_k(\sigma) := \{f : \sigma \rightarrow \mathbb{R}^n : \partial^\alpha f \text{ is absolutely continuous for } |\alpha| \leq k\},$$

$$\mathcal{K}_k(\sigma) := \{g \in \mathcal{A}_k(\sigma) : 0 \leq \partial^\alpha g(x) \leq \partial^\alpha g(y) \text{ for } 0 \leq x \leq y, |\alpha| \leq k\}.$$

$$\mathcal{A}_\infty(\sigma) := \bigcap_{k=1}^{\infty} \mathcal{A}_k(\sigma) \quad \text{and} \quad \mathcal{K}_\infty(\sigma) := \bigcap_{k=1}^{\infty} \mathcal{K}_k(\sigma).$$

Note that Talyor's formula (44) holds for any function $f \in \mathcal{A}_k(\sigma)$.

We can now introduce the notion of a **dominant function** which will be the basis for building the exclusion algorithm used in this thesis.

Definition 4.2.2.2. Let $f \in \mathcal{A}_k(\sigma)$ and $g \in \mathcal{K}_k(\sigma)$. We say that g **dominates** f with order k on σ and write $f(x) \prec_k g(x)$ for $x \in \sigma$ iff the estimates

$$|\partial^\alpha f(x)| \leq \partial^\alpha g(|x|) \text{ for } x \in \sigma$$

hold for all $x \in \sigma$ and $|\alpha| \leq k$.

If $f \in \mathcal{A}_\infty(\sigma)$ and $g \in \mathcal{K}_\infty(\sigma)$, then $f(x) \prec_\infty g(x)$ for $x \in \sigma$ means that $f(x) \prec_k g(x)$ for $x \in \sigma$ and all $k \geq 0$.

Note 4.2.2.3. $f(x) \prec_k g(x)$ for $x \in \sigma$ implies that $f(x) \prec_q g(x)$ for $x \in \tau$ for any $\tau \subset \sigma$ and $q \leq k$. From now on we will try to use the notation $f \prec_k g$ instead of $f(x) \prec_k g(x)$ if there is no ambiguity about the underlying interval.

A connection between dominant functions and the relation defined in the definition 4.2.1.6 is given by the following theorem.

Theorem 4.2.2.4. Let $f(x) = \sum_\alpha f_\alpha x^\alpha$ and $g(x) = \sum_\alpha g_\alpha x^\alpha$ be two power series convergent on an interval $\sigma \subset \mathbb{R}^n$ which contains the origin. Then

$$f \prec_\infty g \iff f \prec\prec g.$$

The following examples point out the differences between the various estimates.

Example 4.2.2.5.

- If $g \in \mathcal{K}_k$ then $g \prec_k g$. This includes examples such as $\exp(m+x) \prec\prec \exp(m+x)$, and $\tan(x) \prec\prec \tan(x)$ for $|x| < \frac{\pi}{2}$.
- $\sin(x) \prec\prec \sinh(x)$, but $\sin(x) \prec_1 x$, $\sin(x) \prec_2 x + \frac{1}{2}x^2$, $\sin(x) \prec_3 x + \frac{1}{6}x^3$.
- $\cos(x) \prec\prec \cosh(x)$, but $\cos(x) \prec_1 1+x$, $\cos(x) \prec_2 1+\frac{1}{2}x^2$, $\cos(x) \prec_3 1+\frac{1}{2}x^2+\frac{1}{6}x^3$.
- $\log(1+x) \prec\prec -\log(1-x)$, but $\log(1+x) \prec_3 x + \frac{1}{2}x^2 + \frac{1}{3}x^3$ for $|x| < 1$.
- $\sin(m+x) \prec\prec \sinh(|m|+x)$, but $\sin(m+x) \prec_2 |\sin(m)| + |\cos(m)|x + \frac{1}{2}x^2$.

The following theorem consists of a list of rules that can be used as a tool to generate dominant functions, in much the same way as rules about differentiation are used as a tool to generate derivatives (see the examples following the theorem).

Theorem 4.2.2.6.

1. If $f \prec_k g$, then $f(m+x) \prec_k g(|m|+x)$.
2. If $f \prec_1 g$, then $|f| \prec_1 g$.
3. If $f \prec_k g$, then $\lambda f \prec_k |\lambda|g$, for any $\lambda \in \mathbb{R}$.
4. If $f_i \prec_k g_i$, $i = 1 : q$, then $\sum_i f_i \prec_k \sum_i g_i$.
5. If $f_i \prec_k g_i$, $i = 1 : q$, then $\prod_i f_i \prec_k \prod_i g_i$.
6. Let $f \prec_k g$ and $f_i \prec_k g_i$, $i = 1 : n$. Set $F = f(f_1, \dots, f_n)$ and $G = g(g_1, \dots, g_n)$. Then $F \prec_k G$.

Example 4.2.2.7.

- $e^{|\sin(m+x)|} \prec_1 e^{|\sin(m)|+x}$ since $e^t \prec\prec e^t$ and $\sin(m+x) \prec_1 |\sin(m)|+x$
- $\sin(x^2) \cos(y-2z) \prec_3 \left(x^2 + \frac{1}{6}(x^2)^3\right) \left(1 + \frac{1}{2}(y+2z)^2 + \frac{1}{6}(y+2z)^3\right)$
- $\frac{1}{1+\frac{1}{9}\cos(x)} \prec_2 \frac{1}{1-\frac{1}{9}(1+\frac{1}{2}x^2)}$ for $x \in [-2\sqrt{5}, 4]$ since $\frac{1}{1+t} \prec\prec \frac{1}{1-t}$ for $|t| < 1$ and $\cos(x) \prec_2 1 + \frac{1}{2}x^2$

4.2.3 Exclusion Tests Using Dominant Functions

Theorem 4.2.3.1. Let $\sigma \subset \mathbb{R}^n$ be an interval, and $q > 0$ be an integer. Let $f(m_\sigma + x) \prec_q g(x)$ for $|x| \leq r_\sigma$. Then

$$|f(m_\sigma)| \leq g(r_\sigma) - g(0) - \sum_{0 < |\alpha| < q} \underbrace{(\partial^\alpha g(0) - |\partial^\alpha f(m_\sigma)|)}_{\geq 0} r_\sigma^\alpha \quad (45)$$

is an exclusion test for f on σ .

Using this theorem, one can summarize the possible exclusion tests.

Corollary 4.2.3.2. *Let $\sigma \subset \mathbb{R}^n$ be an interval, and $q > 0$ be an integer. Let $f \prec_q g$ on σ .*

Then

$$|f(m_\sigma)| \leq g(|m_\sigma| + r_\sigma) - g(|m_\sigma|) - \sum_{0 < |\alpha| < q} \underbrace{(\partial^\alpha g(|m_\sigma|) - |\partial^\alpha f(m_\sigma)|)}_{\geq 0} r_\sigma^\alpha \quad (46)$$

is an exclusion test for f on σ .

Corollary 4.2.3.3 (Lipschitz Constants for f). *Let $\sigma \subset \mathbb{R}^n$ be an interval, let $f \in \mathcal{A}_1(\sigma)$, and consider Lipschitz constants*

$$C_\alpha \geq \sup_{y \in \sigma} |\partial^\alpha f(y)| \text{ for } |\alpha| = 1.$$

Then

$$|f(m_\sigma)| \leq \sum_{|\alpha|=1} C_\alpha r_\sigma^\alpha \quad (47)$$

is an exclusion test for f on σ .

Corollary 4.2.3.4 (Lipschitz Constants for f'). *Let $\sigma \subset \mathbb{R}^n$ be an interval, let $f \in \mathcal{A}_2(\sigma)$, and consider Lipschitz constants*

$$C_\beta \geq \sup_{y \in \sigma} |\partial^\beta f(y)| \text{ for } |\beta| = 2.$$

Then

$$|f(m_\sigma)| \leq \sum_{|\alpha|=1} |\partial^\alpha f(m_\sigma)| r_\sigma^\alpha + \sum_{|\beta|=2} C_\beta r_\sigma^\beta \quad (48)$$

is an exclusion test for f on σ .

Note 4.2.3.5. The terms inside the summation sign in (45) and (46) are nonnegative, and therefore the test tightens as q increases. To increase the efficiency of the implementation, one would successively apply the test for $q = 1, \dots, q_0$ (for some given q_0) and discard the intervals as soon as the test fails. Note also that for $q = 1$, the test (46) reduces to the one given in (43); however, instead of requiring $f \prec\prec g$, we only need to require $f \prec_1 g$ in this case.

4.3 The Solutions for the Bratu Problem

Consider again the Bratu Problem from (38)

$$\begin{aligned} u_{xx} + \lambda \exp(u) &= 0 \quad \text{on } [0,1] \\ u(0) &= u(1) = 0. \end{aligned} \tag{49}$$

The homotopy function which gives a mesh refinement in continuous deformation for this problem is

$$H_{N+1}(u_1, u_2, \dots, u_{N+1}, \lambda, t) :=$$

$$\begin{bmatrix} u_0 - 2u_1 + u_2 & - & \lambda h(t)^2 \exp(u_1) \\ & \vdots & \\ u_{N-2} - 2u_{N-1} + u_N & - & \lambda h(t)^2 \exp(u_{N-1}) \\ u_{N-1} - 2u_N + U_{N+1}(t) & - & \lambda h(t)^2 \exp(u_N) \\ u_N - 2u_{N+1} + U_{N+2}(t) & - & \lambda h(t)^2 \exp(u_{N+1}) \end{bmatrix} \tag{50}$$

where

$$\begin{cases} x_i(t) &:= ih(t), & \forall i = 1, \dots, N+1 \\ u_0 &:= 0 \\ h(t) &:= t \frac{1}{N+1} + (1-t) \frac{1}{N+2} \\ U_{N+1}(t) &:= (1-t)u_{N+1} \\ U_{N+2}(t) &:= 0 \end{cases} \tag{51}$$

The starting points u_{N+1} for $H_{N+1}(u_1, u_2, \dots, u_{N+1}, \lambda, 1) = 0$ can be obtained by solving for u the following equation

$$\exp(u) - au + b = 0, \tag{52}$$

where a and b are constants: $a = 2(N+1)^2/\lambda$, $b = u_N(N+1)^2/\lambda$.

As we saw in section 4.1, our homotopy numerical continuation worked well for $\lambda \in [0, 2.1]$, but (52) had no real solutions for $\lambda \in [2.1, \lambda_*]$ and therefore we had to start looking

for the complex ones. Considering $u = u_{\Re} + iu_{\Im}$ and $b = b_{\Re} + ib_{\Im}$, (52) can be rewritten as

$$\begin{aligned} \exp(u_{\Re}) \cos(u_{\Im}) - au_{\Re} + b_{\Re} &= 0 \\ \exp(u_{\Re}) \sin(u_{\Im}) - au_{\Im} + b_{\Im} &= 0 \end{aligned} \quad (53)$$

and now, we can use our exclusion test from (46) to solve this system.

Let's take

$$F(x, y) = \begin{bmatrix} \exp(x) \cos(y) - ax + b_{\Re} \\ \exp(x) \sin(y) - ay + b_{\Im} \end{bmatrix} \quad (54)$$

and

$$G(x, y) = \begin{bmatrix} \left(1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{\exp(m_{\sigma}^1 + r_{\sigma}^1)}{120} x^5\right) \left(1 + \frac{y^2}{2} + \frac{y^4}{24} + \frac{y^5}{120}\right) + |a|x + |b_{\Re}| \\ \left(1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{\exp(m_{\sigma}^1 + r_{\sigma}^1)}{120} x^5\right) \left(y + \frac{y^3}{6} + \frac{y^5}{120}\right) + |a|y + |b_{\Im}| \end{bmatrix}$$

where $\sigma = [m_{\sigma} - r_{\sigma}, m_{\sigma} + r_{\sigma}] \subset \mathbb{R}^2$ is the box interval in which we are searching for zeros of F , $m_{\sigma} = (m_{\sigma}^1, m_{\sigma}^2)$ and $r_{\sigma} = (r_{\sigma}^1, r_{\sigma}^2)$ are the middle point and the radius of σ , respectively. One can easily check that $F \prec_5 G$ on any interval $\sigma = [m_{\sigma} - r_{\sigma}, m_{\sigma} + r_{\sigma}] \subset \mathbb{R}^2$. We can apply now our homotopy numerical toolbox to solve (50) for $\lambda = 3$. We will use the exclusion test (46) with $q = 5$ for the functions above until both components of the radii of the generated box-intervals in Γ_l are less than $\epsilon = 0.1$. The results we obtained are shown in Tables 3 and 4.

n	$SOLS(n)$	$REAL(n)$
1	6	0
2	35	1
3	210	2
4	1259	3
5	7547	3

Table 3: The number of solutions for $u_{xx} + \lambda \exp(u) = 0$ with zero Dirichlet boundary conditions on $[0, 1]$ for $\lambda = 3$. Homotopy with continuation in t was used to track them from $t = 1$ to $t = 0$; the exclusion algorithm (46) was used to solve (53) in the starting box interval given by $m_{\sigma} = (10, 0)$ and $r_{\sigma} = (10, 15)$.

n	$SOLS(n)$	$REAL(n)$
1	2	0
2	3	1
3	6	2
4	11	3
5	19	3
6	35	3
7	68	3
8	133	3
9	265	5
10	527	5

Table 4: The number of solutions for $u_{xx} + \lambda \exp(u) = 0$ with zero Dirichlet boundary conditions on $[0, 1]$ for $\lambda = 3$. Homotopy with continuation in t was used to track them from $t = 1$ to $t = 0$; the exclusion algorithm (46) was used to solve (53) in the starting box-interval given by $m_\sigma = (5, 0)$ and $r_\sigma = (5, 5)$.

Remark 4.3.1. One needs to realize that the exclusion algorithm process can be very costly and it is applied in our toolbox often (every time we add a new interior point and for every previous solution). For example, in Table 3, for adding one more point to the already existing four points implies using the exclusion algorithm 1259 times to solve equations of the form (53). Also, to reach the goal of $\|r_\sigma\| \leq \epsilon$, for any $\sigma \in \Gamma_l$, we needed $l = 8$ bisection levels each time we used the exclusion algorithm. Below is an example of how the number of intervals $\sigma \in \Gamma_l$ modifies as l increases when exclusion algorithm is used to solve (53) when a fifth point is added to the already existing four interior points.

the bisection level l	0	1	2	3	4	5	6	7	8
the number of intervals σ obtained	1	4	16	64	247	870	1534	14	9

Table 5: The number of intervals obtained at each bisection level for a time when exclusion algorithm was used to obtain the results from Table 3.

Out of these small nine box-intervals, we did obtain at that moment six different solutions for (53). Therefore, it is not a very good idea to pick a big initial box, unless it is necessary.

Remark 4.3.2. For each $n = 1, \dots, 5$, all the solutions found in Table 4 have also been present in Table 3; this was expected since the box-interval chosen for the exclusion algorithm in Table 3 was bigger and included the one used in Table 4. Out of these real solutions (three for Table 3 for $n = 5$ and five for Table 4 for $n = 10$) only two of them were able to be refined to more interior points using a process similar with the one explained in section 2.6. For the other ones, the Newton process did not converge. Below is a plot for these two solutions refined to 1407 interior points. The value of the peaks for these two symmetric solutions are 0.6401 and 1.975 respectively.

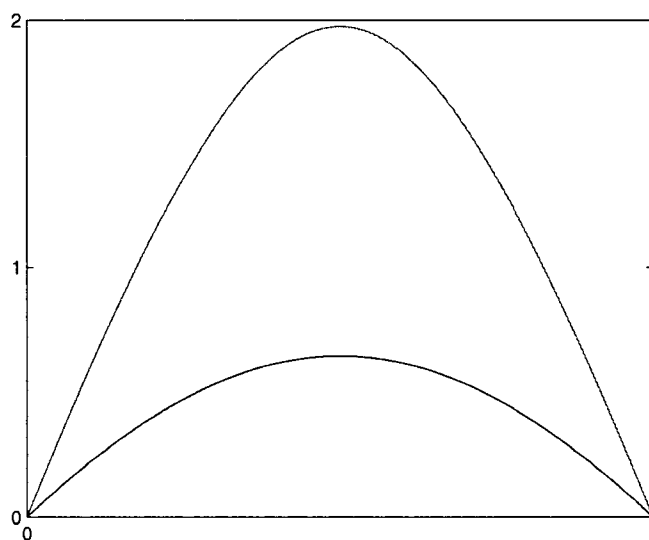


Figure 25: The two real solutions for the Bratu problem (49) refined from a mesh with 10 interior points to one with 1407 interior points.

CHAPTER V

CONCLUSIONS

I will start this paragraph by thanking again to my two advisors for their help, ideas, guidance, patience and encouragement throughout my research at CSU. I have learned a lot from them not only from a theoretical point of view, but from a practical one also, as I started building my toolbox with Matlab codes.

In this chapter we will summarize the work we have accomplished, give some conclusions, and present some extensions, directions and goals.

We started our work by trying to generalize to higher dimensions what E. Allgower and the other authors in [1] accomplished in 1-dimension, namely finding all the solutions of a second order ODE with a polynomial nonlinearity in the right hand side. A summary of their work in this direction was provided in section 1.2. The first problem we had to address was the construction of new meshes. We restricted ourselves to a rectangular domain in 2-dimensions and built a toolbox that helps approximate the derivatives that appear in the Laplace operator. Some calculation and results were presented in the sections 2.3 and 2.4. We then tried to solve the problem (31) for $\lambda = 1$ using homotopy with continuation in t (the homotopy parameter) path tracking. After getting only one real solution on a mesh with 4×4 interior points, we refined it and did the classic arclength continuation. Not too surprisingly, we realized that we were supposed to obtain two real solutions, as the bifurcation diagram shows also. This, together with the fact that we were getting sometimes multiple solutions for our homotopy at $t = 0$ from different initial ones at $t = 1$ directed us toward using arclength continuation for our path tracking. There were some problems initially because of different turning points, but they mostly vanished when we generalized two results from [11] which enabled us to switch branches efficiently and still reach the

goal $t = 0$. Using this method, we were able to obtain the maximum number of solutions for our homotopy at each step (this number is given by Bezout's theorem). For the mesh with 4×4 interior points, we obtained exactly two real solutions out of a total of 2^{16} . With this toolbox, we were also able to solve McKenna's conjecture [5]. We went further and observed that there are really no other bifurcations from the branches we were following for $\lambda < 8000$.

The next interesting step was to try to generalize our problem for a non-polynomial nonlinearity. We knew that our algorithm would work nicely for any kind of function f from (5) for example, if we knew how to solve an equation like (8) for all real and complex solutions. That's when exclusion algorithms came into our help. The only restriction was that we had to seek solutions in a box interval; however this is not so bad since we know that such an equation may have infinitely many solutions. We expanded our toolbox to this case also, and solved Bratu problem in 1-dimension. The results were explained in Chapter IV.

It is clear now that one goal for the future is to expand our methods for other kinds of domains. In both problems (31) and McKenna's conjecture for square domains, we found two solutions that had all the symmetries which the square has. McKenna's problem had two more solutions that that did not have all the symmetries. How many solutions would these two problems have on an L-shape domain for example?

There are only a few results about complex solutions! Can the properties of real solutions be extended to complex ones? We don't really know the answer yet, but it is possible to be 'yes'. The reason we are inclined to say 'yes' lies in the fact explained below.

i) After finding the numerical solutions of (33) on a grid with 4×4 interior points, we tried to refine them to a coarser mesh (79×79), and we did that in four steps, using the algorithm explained in section 2.6: we first refine them to a 9×9 mesh grid, then to 19×19 , 39×39 , and 79×79 in the end.

ii) We have observed that out of all solutions, the two real ones and the 4 complex ones

we discussed in section 2.7 were the only ones for which the peak did not change drastically (in fact, it almost did not change at all in the last two steps of our refining algorithm). Further these solutions satisfied the famous result of Gidas, Ni, Nirenberg Theorem (34).

iii) For the other complex solutions, the changes were major as we tried to refine them, and there were probably spurious solutions; they also did not satisfy the symmetry result (34).

It is important to remark that at this time, we cannot really throw away complex solutions found for example at the end of solving the homotopy after introducing a new point. We have seen in Chapter IV that one of the two real solutions for (49) with λ near the bifurcation value is born only after introducing the 5th interior point. Since the number of solutions grows so quickly with the addition of interior points, it would be nice if we can develop some theoretical filters to discard spurious solutions during our algorithm.

One may also envision generalizing the differential operator and the boundary conditions. Considering other discretization methods, such as finite element methods, can be also taken into consideration in the future. The matter of introducing a small number of elements raises other issues which would be worthy of investigation.

We conclude by emphasizing again that the methods presented in this thesis are not meant to provide fast and accurate methods for solving PDE's. Rather, the idea is to obtain reliable information about the number of solutions as well as qualitative properties of them. However, the approximations we obtain can definitely be used as starting values to get more accurate solutions on finer meshes with methods as the one presented in section 2.6.

CHAPTER VI

APPENDICES

6.1 Appendix A: Mountain Pass Algorithm

In this paragraph, we will present some background about the Mountain Pass Algorithm used by the authors in [5] to find the four solutions to the McKenna's conjecture.

The Mountain Pass Theorem was introduced in 1973 by Ambrosetti and Rabinowitz. By this method, one can establish the existence of a critical point u of a functional F , critical point which is not an extremum point of F , and has the property that in any neighborhood of u there are two points v and w with $F(v) < F(u) < F(w)$. Such a critical point is called a **saddle point** of F .

Definition 6.1.1. Let X be a Banach space. A functional $F \in C^1(X)$ is said to satisfy the **Palais-Smale condition**, for short the (PS) condition, if for any sequence of elements $u_k \in X$ for which

$$F(u_k) \longrightarrow \mu \in \mathbb{R}, \quad F'(u_k) \longrightarrow 0 \quad (55)$$

as $k \longrightarrow \infty$, there exists a convergent subsequence.

Theorem 6.1.2 (Ambrosetti-Rabinowitz). Let X be a Banach space and $F \in C^1(X)$. Suppose there exist $u_0, u_1 \in X$ and r with $|u_0| < r < |u_1|$ such that

$$\max\{F(u_0), F(u_1)\} < \inf\{F(u) : u \in X, |u| = r\}.$$

Let

$$\Gamma = \{\gamma \in C([0, 1]; X) : \gamma(0) = u_0, \gamma(1) = u_1\} \quad (56)$$

and

$$c = \inf_{\gamma \in \Gamma} \max_{t \in [0, 1]} F(\gamma(t)). \quad (57)$$

Then, there exists a sequence of elements $u_k \in X$ such that

$$F(u_k) \longrightarrow c, \quad F'(u_k) \longrightarrow 0 \quad \text{as} \quad k \longrightarrow \infty.$$

If, in addition, F satisfies the (PS) condition, then there exists an element $u \in X \setminus \{u_0, u_1\}$ with

$$F(u) = c, \quad F'(u) = 0. \quad (58)$$

Remark 6.1.3. In the previous theorem, Γ represents the set of all continuous paths joining u_0 and u_1 . In other words, the Mountain Pass Theorem says that if we are at the point u_0 of altitude $F(u_0)$ located in a plateau surrounded by high mountains, and we are looking to reach to a point u_1 of altitude $F(u_1)$ over the other side of the mountains, we can always find a path going from u_0 to u_1 through a mountain pass. To find a mountain pass, we have to choose a path which mounts the least.

Remark 6.1.4. In [26], one can also find Schechter's version of the Mountain Pass Theorem which guarantees the existence of a critical point for a nonlinear functional in a bounded region of the space.

Introduced in [27] for the study of numerical solutions for elliptic boundary value problems, the Mountain Pass Algorithm has often been used since then for a variety of other variational problems ([28, 29, 30]).

The main idea is an implementation of the naive proof of this theorem. First, one associates a functional $F(u)$ (usually nonlinear) such that the solutions of the semilinear elliptic equation correspond to the critical points of this functional. Then, starting with a large finite dimensional approximating space, one constructs a piecewise linear path (initially a straight line) joining an already known critical point u_* which is a local minimum to a suitable chosen point e with $F(e) < F(u_*)$.

For example, for

$$\begin{aligned} \Delta u + f(u) &= h(x) + s\phi_1(x) \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \partial\Omega \end{aligned} \quad (59)$$

where ϕ_1 is the first (positive) eigenfunction of the Laplacian, one can derive via Green's Theorem an associated functional

$$F : H_0^1(\Omega) \rightarrow \mathbb{R} \quad F(u) = \int_{\Omega} \left(\frac{1}{2} |\nabla u|^2 - \widehat{f}(u) + s\phi_1 u + h(x)u \right) dx \quad (60)$$

where \widehat{f} denotes the primitive of f . It is known [31] that the weak solutions of (59) correspond to the critical points of this functional. It was also shown in [32] that (59) has exactly two solutions for large positive s if the derivative of the convex function f has limits at $+\infty$ and $-\infty$, namely, $f'(+\infty)$ and $f'(-\infty)$, and if

$$0 < f'(-\infty) < \lambda_1 < f'(+\infty) < \lambda_2.$$

Then, in a more general setting, Lazer and McKenna have shown in [33] that (59) has *at least* three, and generically four solutions if

$$-\infty < f'(-\infty) < \lambda_1 < \lambda_{2n} < f'(+\infty) < \lambda_2 < \lambda_{2n+1}$$

where, in addition, it was assumed that λ_2 was of odd multiplicity. Two of these solutions are asymptotically (as $s \rightarrow \infty$) linear, and were approximated by $u^* = s\phi_1/(f'(+\infty) - \lambda_1)$ and $u_* = s\phi_1/(f'(-\infty) - \lambda_1)$. Since u_* is almost a linear solution, it was shown that it is also a local minimum for the associated functional (60) on the function space $H_0^{1,2}(\Omega)$.

We may apply the Mountain Pass Theorem to (60) by observing that the functional F is in $\mathcal{C}^1(X)$, where $X = H_0^{1,2}(\Omega)$ and Ω is a nice bounded region in \mathbb{R}^2 . One can now construct a piecewise linear path joining u_* to a suitable chosen point e with $F(e) < F(u_*)$. For McKenna's conjecture, that we can always choose such an e is clear from the nature of the nonlinearity u^2 and the associated functional $F(u)$ with $\widehat{f}(u) = u^3/3$. Observe also that $F(s\phi_1) \rightarrow -\infty$ as $s \rightarrow \infty$, and therefore, by going sufficiently far in the direction of any fixed positive function u , we can always find such an e .

One then searches along discrete points on the path for the point at which the functional F is maximized. After locating it, one alters that piece of the path by moving that node in the direction of *steepest descent*. One iterates on that procedure, taking appropriate

safeguards to ensure that the integrity of the piecewise linear path remains intact, without too much stretching between successive points. Eventually, the method converges to a saddle point. For more details about Mountain Pass Algorithm, one can also refer to [5].

6.2 *Appendix B: Manual For Our Toolbox*

In this appendix we will present some explanations and some instructions that will allow the reader to use our toolbox for the problems we presented in the thesis as well as for other similar ones.

The folder ‘HomLabFiles contains’ the files of the HomLab toolbox created by the authors in [2]. It is a suite of scripts and functions for the Matlab environment designed as an easy entry into the use of polynomial continuation and, for the experienced user, as a platform for experimental development of new methods. A user’s guide for it was written in the Appendix C from [2].

In the folder ‘DiscretizationForLaplace’, one can find three types of files. One of them is **SolForm.m** where we describe how the solution u of the discretized system (26) looks when a new point is introduced on a *row* or a *column* of points (see figures 4 and 5 for more details).

The other two types of m-files have the names starting with **Stiff** or **boundary**. In the m-files for which their names start with **Stiff**, we are building the components A_{xx} and A_{yy} of the stiffness matrix A from (26); in the m-files for which their names start with **boundary**, we are building the vectors \vec{b}_{xx} and \vec{b}_{yy} arising from the boundary values of the discretization of Δu from (25) as presented in section 2.2. It is important to remark that all these components of the homotopy H_{N+1} from (30) are sparse and constructed with block matrices and vectors as presented in section 2.4. These files can be used for any kind of 2-D problem of the form (25) where the domain Ω is a rectangle of the form

$$\Omega = [a, b] \times [c, d] \subset \mathbb{R}^2.$$

For example, in **Stiff_yy_line.m**, we construct the matrix A_{yy} when the new point is introduced on a row L (the form of this matrix can be seen in section 2.4). To call this function, one can use a command such as:

$$A_{yy} = \text{Stiff_yy_line}(L, t, N, M, a, b, c, d);$$

where

- L - is the row where the new point was introduced;
- t - is the homotopy parameter;
- N - is the number of points on the rows $(L + 1), \dots, M$. Hence, rows $1, \dots, L$ each has $(N + 1)$ points;
- M - is the number of points on each column;
- a, b, c, d - are the values corresponding to the rectangular domain of the problem,

$$\Omega = [a, b] \times [c, d].$$

If the new point is introduced on a *column* instead of a *row*, then the files we will use have the word **col** instead of **line** at the end of their names (**boundary_xx_col.m**, for example).

The folder ‘ResultsFor_p_1_0_1’ contains the solutions of the 2-D problem (33)

$$\begin{aligned} \Delta u &= -\lambda(1 + u^2) & \text{on } \Omega &= [0, 1]^2 \\ u(0, y) &= u(1, y) = u(x, 0) = u(x, 1) = 0, \forall x, y \in [0, 1] \end{aligned}$$

for $\lambda = 1$. As presented in section 2.7.1, we found all 2^N solutions of the homotopy H_N using a tracker based on arclength continuation and saved them in the subfolder ‘p_1_0_1_ArcLContTrack_AllSolsUntil4x4IntPoints’; the results were summarized in the

Table 2 from section 2.7.1. The two real solutions (out of a total of 2^{16}) obtained on 4×4 interior points were refined to a coarser mesh-grid (39×39) using the method presented in section 2.6. Using numerical arclength continuation and starting with any of these two refined solutions, we immediately built the bifurcation diagram drawn in Figure 19 (it is saved in the ‘p_1_0_1_ArcLContTrack_SolsForBifDiagram’ subfolder).

The initial results of this problem that were found using a tracker based on continuation in t (the homotopy parameter) are saved in the folder

‘UsingContIn_t_TrackerWithGammaTrickWhenNecessary’

and summarized in Table 1, section 2.7.1.

The numerical results for the McKenna’s conjecture (presented in Chapter III) can be found now in the folder ‘ResultsForMcKennaProblem’. One can also take a look at the eigenvalues of the Jacobian saved here as we constructed the bifurcation diagram and observe the turning point, the symmetry breaking bifurcation and the fact that there are no other bifurcations for big values of λ (we went with λ until around 8000). We can also watch some movies with the changes in the solutions as λ follows the bifurcation diagram. It is easy to observe in this case the symmetry breaking bifurcation point: as λ approaches $\lambda_S \approx 587.7$, any of the two solutions from the right side in Figure 21 tends to become fully symmetric and positive, as the one from the top left side of this figure.

The numerical results for the 1-D Bratu Problem (49) using homotopy continuation combined with exclusion algorithm were presented in section 4.3 and can be found now in the folder ‘ResultsFor1DBratu_HomotWExclAlg’.

The ‘RunningFiles’ folder contains the main files that one can use to run for the problems considered in this thesis as well as other similar ones. For example, to solve McKenna’s

conjecture (36) from the beginning using homotopy with arclength continuation, one would call the file

yk_main_2d_mckenna_Arclength.m

and to solve the problem (33), one would call the file

yk_main_2d_Mmodif_FullAllArclength.m

Note that these files are built for a more general class of problems. When such files are run, one can choose

- a different rectangular domain by entering other values for a, b, c, d than 0, 1, 0, 1;
- a different polynomial right hand side by entering other values for the coefficients rather than the vector [1, 0, 1] which stands for $u^2 + 0u + 1$; for example, for $f(u) = u^3$, one would enter [1, 0, 0, 0] from the keyboard when asked to input the coefficients of the polynomial right hand side;
- different value for λ .

In fact, one can also solve (59) which is a generalization of McKenna's conjecture. For this, one should first modify correspondingly the file **func.f.m**, and then run **yk_main_2d_mckenna_Arclength.m**. Below is the Matlab file **func.f.m** for McKenna's conjecture.

```
function out = func_f(x,y);
% The function f from:
%   Laplace(u) + lambda*p(u) + f(x,y) = 0
out = -800*sin(pi*x)*sin(pi*y);
```

Since the solutions we get are usually on a crude mesh, we can refine the ones which interest us to a coarser grid by calling one of the **refinesol.m** files. Then, we can run the **regular_arclength** files to start the arclength continuation and find the bifurcation diagram.

For example, after refining one of the solutions for McKenna's conjecture to a mesh with a 31×31 interior points for $\lambda = 800$, we ran the **mckenna_regular_arclengthModif.m** file and found the bifurcation diagram presented in Figure 21. We need to remark that we used only 2^{nd} order discretization for our derivatives. If one saves these solutions found at each step of our arclength continuation as columns of a matrix, then, by running the **sol_movie_arclength.m** file we can see a movie with the evolution of the initial solution as the bifurcation parameter λ changes. Running the movie for any of the two non-fully symmetric solutions we found, one can easily observe the symmetry breaking bifurcation point, as explained before.

Note that if we would like to run **mckenna_regular_arclengthModif.m**, but for the more general problem (59), we first modify accordingly the file **func_f_lambda.m**. For the McKenna's conjecture, this file looks like

```
function out = func_f_lambda(x,y,lambda);
% The function f from:
%   Laplace(u) + lambda*p(u) + f(x,y) = 0
out = -lambda*sin(pi*x)*sin(pi*y);
```

The exclusion test (46) presented in section 4.2.3 is implemented in the **exclusion_18.m** file. To be able to run this file, one needs to provide two Matlab files corresponding to the functions f and g , files in which we have to include all the partial derivatives needed (recall the exclusion test (46)). For example, the Matlab files corresponding to the functions F and G (see (54)) for the 1-dimensional Bratu problem (49) were created using the Maple 9 file **ForBratu.mws**. The file corresponding to F (**BratuFunctions_F_Filea.m**) will be presented at the end of this appendix.

To use the **exclusion_18.m** file for the Bratu problem, one would call

$$\text{excl} = \text{exclusion_18}(\text{ab}, \text{eps}, \text{mm}, \text{nnr}, \text{nni}, \text{qq});$$

where

- ab - represents the box-interval in $\mathbb{R}^d \times \mathbb{R}^d$ where we are looking for the zeros of F ;
- eps - our output intervals in \mathbb{R}^d (where we might have solutions) will have all d components of the radius less than or equal to this eps ;
- mmm, nnr, nni - are the parameters a, b_{\Re}, b_{\Im} , respectively from (54);
- q - represents the maximum order for the derivatives considered in (46).

In section 4.3 we have chosen $ab=[10, 0, 10, 15]$ (and $[5, 0, 5, 5]$ for a second case), $eps=0.1$, and $q=5$.

The output of this file will be a matrix named `excl` in which each row r contains the box-intervals generated by the exclusion algorithm at the r^{th} bisection level (they form the set Γ_l from Remark 4.2.1.3). The last row of this matrix will be the most updated one, having the all the undiscarded intervals of length up to eps . The first element of each row r represents the number of intervals found at the end of the r^{th} bisection level. The next elements of a row represent the miniboxes where the test indicated that we might have a solution.

Once the miniboxes are obtained, we can call the **`find_complex_zeros.bettera.m`** file to find the roots of F inside of them.

To solve 1-dimensional the Bratu problem from the beginning (with the use of homotopy numerical continuation combined with exclusion algorithm), one would call the main file

`yk_main_bratu_complexa.m`

Note also that by calling **`yk_main_bratu.m`**, one would solve the Bratu problem (38), but using a simple exclusion test (monotonicity, (42)) to find all the real solutions of (40). This is indeed a much faster method, but as we saw in the beginning of section 4.1, this worked well only for values of $\lambda \in [0, 2.1]$.

To run our toolbox for different problems similar to the Bratu problem, one needs

- i) to modify the files **funcf.m**, **funcg.m**, **funch.m**;
- ii) to create new files corresponding to the functions f and g for the exclusion test;
- iii) to make some small changes inside of **yk_main_bratu_complexa.m**;
- iv) to run **yk_main_bratu_complexa.m**.

One of the most important files is **tracker_2d_Modify.m** which helps us to follow the path of a solution as the homotopy parameter t goes from 1 to 0. The listing of one version of this file is presented in the next appendix. During our path-tracking, we can pay attention also at the angle between two consecutive tangents. For efficiency, the arclength step should be decreased if this angle is too big, and it should be increased if the angle is too small. One can remark that we have this implemented in our path-tracking file, but we commented it out because it significantly slows down the algorithm!

As mentioned before, in the end of this appendix, we present the Matlab code obtained for the function F (see (54)) using the Maple 9 file **ForBratu.mws**.

```
function out = BratuFunctions_F_File(x,y, mmm,nnnr,nnni, ms,rs, nrderiv)
% This function contains all the functions corresp to F
% (including its derivatives) necessary to apply the test (46).
% The function F might have d real component functions, and
% hence our file will build a function out with this format:
% out(k,i,j), which is the (i,j) derivative of the k'th component of F.
% For example, out(2,3,4) means that this is
% 3'rd deriv in x, the 4'th in y of the 4'th component of F.
% For example, out(2,0,0) means that this is just the 2'th component of F.
% BUT WE KNOW MATLAB DOES NOT LIKE TO WORK WITH 0 AS AN INDICE,
% so we do this convention: out(k,i,j) means now that this is
% the (i-1,j-1) derivative of the k'th component of F. Hence,
% for example, out(3,1,1) is just the 3'rd component of F, and
% out(3,2,4) is the 1'st deriv in x, 3'rd in y of the 3'rd component of F.
```

```
% OBS: You will need also the function G found in BratuFunctions_G_File
% such that  $F \leq_{\{q\}} G$ .
```

```
out( 1, 1, 1) = exp(x)*cos(y)-mmm*x-nnr ;
out( 1, 1, 2) = -exp(x)*sin(y) ;
out( 1, 1, 3) = -exp(x)*cos(y) ;
out( 1, 1, 4) = exp(x)*sin(y) ;
out( 1, 1, 5) = exp(x)*cos(y) ;
out( 1, 1, 6) = -exp(x)*sin(y) ;
out( 1, 2, 1) = exp(x)*cos(y)-mmm ;
out( 1, 2, 2) = -exp(x)*sin(y) ;
out( 1, 2, 3) = -exp(x)*cos(y) ;
out( 1, 2, 4) = exp(x)*sin(y) ;
out( 1, 2, 5) = exp(x)*cos(y) ;
out( 1, 2, 6) = -exp(x)*sin(y) ;
out( 1, 3, 1) = exp(x)*cos(y) ;
out( 1, 3, 2) = -exp(x)*sin(y) ;
out( 1, 3, 3) = -exp(x)*cos(y) ;
out( 1, 3, 4) = exp(x)*sin(y) ;
out( 1, 3, 5) = exp(x)*cos(y) ;
out( 1, 3, 6) = -exp(x)*sin(y) ;
out( 1, 4, 1) = exp(x)*cos(y) ;
out( 1, 4, 2) = -exp(x)*sin(y) ;
out( 1, 4, 3) = -exp(x)*cos(y) ;
out( 1, 4, 4) = exp(x)*sin(y) ;
out( 1, 4, 5) = exp(x)*cos(y) ;
out( 1, 4, 6) = -exp(x)*sin(y) ;
out( 1, 5, 1) = exp(x)*cos(y) ;
out( 1, 5, 2) = -exp(x)*sin(y) ;
out( 1, 5, 3) = -exp(x)*cos(y) ;
out( 1, 5, 4) = exp(x)*sin(y) ;
out( 1, 5, 5) = exp(x)*cos(y) ;
```

```

out( 1, 5, 6) = -exp(x)*sin(y) ;
out( 1, 6, 1) = exp(x)*cos(y) ;
out( 1, 6, 2) = -exp(x)*sin(y) ;
out( 1, 6, 3) = -exp(x)*cos(y) ;
out( 1, 6, 4) = exp(x)*sin(y) ;
out( 1, 6, 5) = exp(x)*cos(y) ;
out( 1, 6, 6) = -exp(x)*sin(y) ;
out( 2, 1, 1) = exp(x)*sin(y)-mmm*y-nnni ;
out( 2, 1, 2) = exp(x)*cos(y)-mmm ;
out( 2, 1, 3) = -exp(x)*sin(y) ;
out( 2, 1, 4) = -exp(x)*cos(y) ;
out( 2, 1, 5) = exp(x)*sin(y) ;
out( 2, 1, 6) = exp(x)*cos(y) ;
out( 2, 2, 1) = exp(x)*sin(y) ;
out( 2, 2, 2) = exp(x)*cos(y) ;
out( 2, 2, 3) = -exp(x)*sin(y) ;
out( 2, 2, 4) = -exp(x)*cos(y) ;
out( 2, 2, 5) = exp(x)*sin(y) ;
out( 2, 2, 6) = exp(x)*cos(y) ;
out( 2, 3, 1) = exp(x)*sin(y) ;
out( 2, 3, 2) = exp(x)*cos(y) ;
out( 2, 3, 3) = -exp(x)*sin(y) ;
out( 2, 3, 4) = -exp(x)*cos(y) ;
out( 2, 3, 5) = exp(x)*sin(y) ;
out( 2, 3, 6) = exp(x)*cos(y) ;
out( 2, 4, 1) = exp(x)*sin(y) ;
out( 2, 4, 2) = exp(x)*cos(y) ;
out( 2, 4, 3) = -exp(x)*sin(y) ;
out( 2, 4, 4) = -exp(x)*cos(y) ;
out( 2, 4, 5) = exp(x)*sin(y) ;
out( 2, 4, 6) = exp(x)*cos(y) ;
out( 2, 5, 1) = exp(x)*sin(y) ;
out( 2, 5, 2) = exp(x)*cos(y) ;

```

```
out( 2, 5, 3) = -exp(x)*sin(y) ;
out( 2, 5, 4) = -exp(x)*cos(y) ;
out( 2, 5, 5) = exp(x)*sin(y) ;
out( 2, 5, 6) = exp(x)*cos(y) ;
out( 2, 6, 1) = exp(x)*sin(y) ;
out( 2, 6, 2) = exp(x)*cos(y) ;
out( 2, 6, 3) = -exp(x)*sin(y) ;
out( 2, 6, 4) = -exp(x)*cos(y) ;
out( 2, 6, 5) = exp(x)*sin(y) ;
out( 2, 6, 6) = exp(x)*cos(y) ;
```

6.3 *Appendix C: A version of a ‘tracker’ file used to solve the problem (33)*

```
function [x,tangent,t,success,stepsize,errsize,nfe]= ...
    tracker_2d_Mmodif(hfun,xinit,tangent,tinit,stepstart,tgoal,epstrack)

% *****
% *||=====||*
% *|| Path Tracker Algorithm ||*
% *||=====||*
% *****
%
%
% INPUTS
% hfun      = string name of homotopy function h(x,t)
%             evaluated as [h,hx,ht]=feval(hfun,x,t).
% xinit     = starting value of x. An approximate solution of h(x,t)=0.
% tangent   = tangent at (x,t). If empty, is computed fresh.
%             If known ahead, passing it in saves a bit of computation.
% tinit     = t value at start.
% stepstart = initial arclength step size to try.
% tgoal     = stopping value for t. We assume tgoal<t.
% epstrack  = path tracking accuracy to be maintained.
%
% OUTPUTS
% x,t       = final solution estimate. If successful, t=tgoal.
% tangent   = tangent at (x,t).
% success   = 1, if successful; 0, if not.
% stepsize  = final step size in t. Can be used as stepstart if we call
%             again for progress along the same path.
% errsize   = last Newton residual.
% nfe       = number of function evaluations used.
```

```

global stepmin maxit maxnfe epstiny
global a b c d pol lambda % L M N

% Some initializations
goodstep = 0; success = 1; nfe = 0;
iter = 0;

% Initialize Newton iteration parameters
tol=1E-6; tolf=1E-9; nitemax=150; ifail=0;

% Initialize arclength parameters
x0 = xinit;
n = size(x0,1);
arc0 = 0;
t0 = tinit;
tau0 = zeros(n,1); sigma0 = 1; theta = 0.5;

% Initialize solution
v0 = xinit; v = xinit;
arcstepmodif = stepstart;

j = 1;
arc = 0;
t = t0;

% Save this initial values in the "old" variables
tau_old = tau0;
sigma_old = sigma0;
arc_old = arc0;
v_old = v0;
t_old = t0;

```

```

while (tgoal < t) & (t <= 1)
    j = j+1;
    arc = arc+abs(arcstepmodif);
% Predictor
    v = v + arcstepmodif*tau0;
    t = t + arcstepmodif*sigma0;

% Corrector
    nite=0; delz=1; f=1;
    while norm(f) > tolf
        nite = nite + 1;
        [f,fz,ft]=feval(hfun,v,t);
        n=size(f,1);
        f(n+1) = theta*tau0'*(v-v0) + (1-theta)*sigma0*(t-t0) - (arc0-arc);
        fz(1:n,n+1) = ft;
        fz(n+1,1:n) = theta*tau0';
        fz(n+1,n+1) = (1-theta)*sigma0;
        delz = fz\f;
        z = [v; t];
        z = z-delz;
        v = z(1:n);
        t = z(n+1);
        fprintf('%3i  %8.4f  %8.4f  %8.4e          %8.4e \n',
            nite, arc, t, norm(delz), norm(f) );
        if nite > nitemax
            fprintf('Maximum number of Newton iterations exceeded\n');
            ifail = 1;
            break
        end
    end
end

if ifail == 0

```

```

    fprintf('Newton iteration converged\n');
    tau_old = tau0;
    sigma_old = sigma0;
    arc_old = arc0;
    v_old = v0;
    t_old = t0;
%   Update tangent vector
    tau0 = -(v-v0)/(arc-arc0);
    sigma0 = -(t-t0)/(arc-arc0);
    normv = norm([tau0;sigma0],2);
    tau0 = tau0/normv;
    sigma0 = sigma0/normv;
%   Update previous solution
    arc0 = arc;
    v0 = v;
    t0 = t;

%   % Modify the arclength step if the angle between the two
%   % consecutive tangents is bigger than 30 degrees or less
%   % than 5 degrees.
%   if j >= 3
%   %costheta=(norm(tau0,2)^2+norm(tau_old,2)^2-norm(tau0-tau_old,2)^2)/
%   %
%   %                                     (2*norm(tau0,2)*norm(tau_old,2));
%   costheta = [tau_old;sigma_old]'*[tau0;sigma0];
%   if costheta < cos(pi/6)
%       % the angle between 2 consec tangents is bigger than 30
%       % degrees, hence restart with the last point and reduce
%       % the step size to half.
%       arc = arc-abs(arcstepmodif);
%       arcstepmodif = arcstepmodif/2;
%       tau0 = tau_old;
%       sigma0 = sigma_old;
%       arc0 = arc_old;

```



```

%      v0      = v_old;
%      t0      = t_old;
%  else
%      if costheta > cos(pi/36)
%          % the angle between 2 consec tangents is less than 5
%          % degrees, hence double the arclength step.
%          arcstepmodif = 2*arcstepmodif;
%      end
%  end
%  end

%      if nite <= 3
%          % Double the arcstep since the number of newton iterations is
%          % very small.
%          arcstepmodif = 2*arcstepmodif;
%      else
%          if nite >= 100
%              % Reduce the arcstep since the number of Newton iterations
%              % is too big.
%              arcstepmodif=.25*arcstepmodif
%          end
%      end

else

    % In this case, ifail=1, which means that the number of Newton
    % iterations exceeded its maximum. Restart with the prev. solution
    % and reduce the arcstep this time!
    tau0 = tau_old;
    sigma0 = sigma_old;
    arc0 = arc_old;
    v0 = v_old;
    t0 = t_old;

```

```

        arcstepmodif = 0.25*arcstepmodif;
    end

    nfe=nfe+iter;
end

if t < 0
    fprintf('We have passed t = 0 directly, after eventually passing');
    fprintf('couple of turning points. We did not reach back t = 1.\n');
    j = j+1;
    step = tgoal - t;
    t = t + step;
    arc = arc - abs(arcstepmodif)*abs(step/(stepstart*sigma_old));
    nite=0; delv=1; f=1;
    while norm(f) > tolf
        nite = nite + 1;
        [f,fv,ft] = feval(hfun,v,t);
        delv = fv\f;
        v = v-delv;
    %fprintf('%3i %8.4f %8.4f %8.4e %8.4e \n',nite,arc,t,norm(delv),norm(f));
        if nite > nitemax
            % We should not really reach this point! But it was implemented
            % just to be sure we did not have a mistake!
            ifail = 1;
            error('Newton iters. exceeded max when we tryied to land on 0!');
        end
    end
    fprintf('We have reached t = 0 directly, after passing couple of ');
    fprintf('turning points. We did not reach back t = 1.\n');
else
    if t > 1

```

```

fprintf('We passed again t = 1.\n');
j = j+1;
step = tinit - t;
t = t + step;
arc = arc - abs(arcstepmodif)*abs(step/(stepstart*sigma_old));
nite=0; delv=1; f=1;
while norm(f) > tolf
    nite = nite + 1;
    [f,fv,ft] = feval(hfun,v,t);
    delv = fv\f;
    v = v-delv;
    fprintf('%3i %8.4f %8.4f %8.4e %8.4e \n',nite,arc,t,norm(delv),norm(f));
    if nite > nitemax
        % We should not really reach this point! But it was
        % implemented just to be sure we did not have a mistake!
        ifail = 1;
        error('Newton iters. exceeded max trying to land on 1!');
    end
end
fprintf('We started at t=1 ended up again at t=1!\n');
end
end
end

```

```

if t == tgoal
    % We've reached the goal t=0; now refine the answer
    x = v;
    iter = 0;
    errsize=1;
    while errsize>1e-12 & iter<10
        iter = iter+1;
        [h,dhx,dht] = feval(hfun,x,t);
    end
end

```

```

        dx = -dhx\h;
        esize = max(abs(dx));
        x = x+dx;
        if esize>2*errsize
            errsize = esize;
            break;
        end;
        errsize = esize;
    end;
    [h,dhx,dht] = feval(hfun,x,t);
    tangent = -dhx\dht;
    nfe = nfe+iter+1;

    %success = arcstepmodif>stepmin & nfe<maxnfe;
    success = nfe<maxnfe;
    stepsize = arcstepmodif;
else
    % We have reached again t = 1 instead of t = tgoal; it means that,
    % during our tracking as t was supposed to go from 1 to 0, we found a
    % turning point. We start back with all the initial data, and this
    % time, when we reach the turning point, we will not continue on our
    % arclength path, but we will jump on the path which will be given in
    % this point by the perpendicular tangent on our normal tangent.

    % Initialize Newton iteration parameters
    tol=1E-6; tolf=1E-9; nitemax=150; ifail=0;

    % Initialize arclength parameters
    x0 = xinit;
    n = size(x0,1);
    arc0 = 0;
    t0 = tinit;
    tau0 = zeros(n,1);  sigma0 = 1;  theta = 0.5;

```

```

% Initialize solution
v0 = xinit; v = xinit;
arcstepmodif = stepstart;

j = 1;
arc = 0;
t = tinit;

% Save this initial values in the "old" variables
tau_old = tau0;
sigma_old = sigma0;
arc_old = arc0;
v_old = v0;
t_old = t0;

while sign(real(sigma0)*real(sigma_old)) == 1
    j = j+1;
    arc = arc+abs(arcstepmodif);
    % Predictor
    v = v + arcstepmodif*tau0;
    t = t + arcstepmodif*sigma0

    % Corrector
    nite=0; delz=1; f=1;
    while norm(f) > tolf
        nite = nite + 1;
        [f,fz,ft]=feval(hfun,v,t);
        n=size(f,1);
        f(n+1) = theta*tau0'*(v-v0)+(1-theta)*sigma0*(t-t0)-(arc0-arc);
        fz(1:n,n+1) = ft;
        fz(n+1,1:n) = theta*tau0';
        fz(n+1,n+1) = (1-theta)*sigma0;

```

```

delz = fz\f;
z = [v; t];
z=z-delz;
v = z(1:n);
t = z(n+1);
fprintf('%3i %8.4f %8.4f %8.4e %8.4e \n',
%       nite, arc, t, norm(delz), norm(f) );
if nite > nitemax
    fprintf('Maximum number of Newton iterations exceeded\n');
    ifail = 1;
    break
end
end
if ifail == 0
    fprintf('Newton iteration converged\n');
    tau_old = tau0;
    sigma_old = sigma0;
    arc_old = arc0;
    v_old = v0;
    t_old = t0;

    %       Update tangent vector
    tau0 = -(v-v0)/(arc-arc0);
    sigma0 = -(t-t0)/(arc-arc0);
    normv = norm([tau0;sigma0],2);
    tau0 = tau0/normv;
    sigma0 = sigma0/normv;
    %       Update previous solution
    arc0 = arc;
    v0 = v;
    t0 = t;

%       % Modify the arclength step if the angle between the two

```

```

%      % consecutive tangents is bigger than 30 degrees or less
%      % than 5 degrees.
%      if j >= 3
%      %costheta=(norm(tau0,2)^2+norm(tau_old,2)^2-norm(tau0-tau_old,2)^2)/
%      %
%      %      (2*norm(tau0,2)*norm(tau_old,2));
%      costheta = [tau_old;sigma_old]'*[tau0;sigma0];
%      if costheta < cos(pi/6)
%          % the angle between 2 consec tangents is bigger than 30
%          % degrees, hence restart with the last point and reduce
%          % the step size to half.
%          arc = arc-abs(arcstepmodif);
%          arcstepmodif = arcstepmodif/2;
%          tau0 = tau_old;
%          sigma0 = sigma_old;
%          arc0 = arc_old;
%          v0 = v_old;
%          t0 = t_old;
%      else
%          if costheta > cos(pi/36)
%              % the angle between 2 consec tangents is less than 5
%              % degrees, hence double the arclength step.
%              arcstepmodif = 2*arcstepmodif;
%          end
%      end
%  end
%  end

%      if nite <= 3
%          % Double the arcstep since the number of newton iterations is
%          % very small.
%          arcstepmodif = 2*arcstepmodif;
%      else
%          if nite >= 100
%              % Reduce the arcstep since the number of Newton iterations

```

```

%           % is too big.
%           arcstepmodif=.25*arcstepmodif
%       end
%   end

    else
% In this case, ifail=1, which means that the number of Newton
% iterations exceeded its maximum. Restart with the prev. solution
% and reduce the arcstep this time!
        tau0 = tau_old;
        sigma0 = sigma_old;
        arc0 = arc_old;
        v0 = v_old;
        t0 = t_old;

        arcstepmodif = 0.25*arcstepmodif;
    end

    nfe=nfe+iter;
end

% We have reached t=t* a turning point. We know that this turning point
% is in fact a bifurcation, hence, using the theory presented in
% chapter I, we can switch the branches. Let's switch now to the
% bifurcating branch!
tau0 = i*tau0;
sigma0 = -sigma0;

while real(t) > 0
    j = j+1;
    arc = arc+abs(arcstepmodif);
    % Predictor
    v = v + arcstepmodif*tau0;

```



```

t = t + arcstepmodif*sigma0

% Corrector
nite=0; delz=1; f=1;
while norm(f) > tolf
    nite = nite + 1;
    [f,fz,ft]=feval(hfun,v,t);
    n=size(f,1);
    f(n+1) = theta*tau0'*(v-v0)+(1-theta)*sigma0*(t-t0)-(arc0-arc);
    fz(1:n,n+1) = ft;
    fz(n+1,1:n) = theta*tau0';
    fz(n+1,n+1) = (1-theta)*sigma0;
    delz = fz\f;
    z = [v; t];
    z=z-delz;
    v = z(1:n);
    t = z(n+1);
    fprintf('%3i  %8.4f  %8.4f  %8.4e          %8.4e \n');
    fprintf('nite, arc,      t,      norm(delz), norm(f) ');
    if nite > nitemax
        fprintf('Maximum number of Newton iterations exceeded\n');
        ifail = 1;
        break
    end
end
if ifail == 0
    fprintf('Newton iteration converged\n');
    tau_old = tau0;
    sigma_old = sigma0;
    arc_old = arc0;
    v_old = v0;
    t_old = t0;

```

```

%      Update tangent vector
tau0    = -(v-v0)/(arc-arc0);
sigma0  = -(t-t0)/(arc-arc0);
normv   = norm([tau0;sigma0],2);
tau0    = tau0/normv;
sigma0  = sigma0/normv;

%      Update previous solution
arc0 = arc;
v0 = v;
t0 = t;

%      % Modify the arclength step if the angle between the two
%      % consecutive tangents is bigger than 30 degrees or less
%      % than 5 degrees.
%      if j >= 3
%      %costheta=(norm(tau0,2)^2+norm(tau_old,2)^2-norm(tau0-tau_old,2)^2)/
%      %
%      %      (2*norm(tau0,2)*norm(tau_old,2));
%      costheta = [tau_old;sigma_old]'*[tau0;sigma0];
%      if costheta < cos(pi/6)
%      % the angle between 2 consec tangents is bigger than 30
%      % degrees, hence restart with the last point and reduce
%      % the step size to half.
%      arc = arc-abs(arcstepmodif);
%      arcstepmodif = arcstepmodif/2;
%      tau0 = tau_old;
%      sigma0 = sigma_old;
%      arc0 = arc_old;
%      v0 = v_old;
%      t0 = t_old;
%      else
%      if costheta > cos(pi/36)
%      % the angle between 2 consec tangents is less than 5
%      % degrees, hence double the arclength step.

```

```

%         arcstepmodif = 2*arcstepmodif;
%     end
% end
% end

%     if nite <= 3
%         % Double the arcstep since the number of newton iterations is
%         % very small.
%         arcstepmodif = 2*arcstepmodif;
%     else
%         if nite >= 100
%             % Reduce the arcstep since the number of Newton iterations
%             % is too big.
%             arcstepmodif=.25*arcstepmodif
%         end
%     end

%     else
% In this case, ifail=1, which means that the number of Newton
% iterations exceeded its maximum. Restart with the prev. solution
% and reduce the arcstep this time!
        tau0 = tau_old;
        sigma0 = sigma_old;
        arc0 = arc_old;
        v0 = v_old;
        t0 = t_old;

        arcstepmodif = 0.25*arcstepmodif;
    end

    nfe=nfe+iter;
end

```

```

% We have reached t<tgoal!

if real(t) < 0
    j = j+1;
    step = tgoal - t;
    t = t + step;
    arc = arc - abs(arcstepmodif)*abs(step/(stepstart*sigma_old));
    nite=0; delv=1; f=1;
    %while norm(delv) > tol
    while norm(f) > tolf
        nite = nite + 1;
        [f,fv,ft] = feval(hfun,v,t);
        delv = fv\f;
        v = v-delv;
        %fprintf('%3i %8.4f %8.4f %8.4e %8.4e \n');
        %fprintf('nite, arc, t, norm(delv), norm(f) ');
        if nite > nitemax
            % We should not really reach this point! But it was
            % implemented just to be sure we did not have a mistake!
            ifail = 1;
            error('Newton iters. exceeded max trying to land on 1!');
        end
    end
end
end

end
end

```

```

if t == tgoal
    % We've reached the goal t; now refine the answer
    x = v;
    iter = 0;
    errsize=1;
    while errsize>1e-12 & iter<10,
        iter = iter+1;
        [h,dhx,dht] = feval(hfun,x,t);
        dx = -dhx\h;
        esize = max(abs(dx));
        x = x+dx;
        if esize>2*errsize
            errsize = esize;
            break;
        end;
        errsize = esize;
    end;
    [h,dhx,dht] = feval(hfun,x,t);
    tangent = -dhx\dht;
    nfe = nfe+iter+1;

    %success = arcstepmodif>stepmin & nfe<maxnfe;!!!!!!!!!!!!!!
    success = nfe<maxnfe;
    stepsize = arcstepmodif;
else
    % I am not expecting to reach here, but for any case, I consider also
    % this case to know if I have to debug it!
    error('Did not expect this error! I should try to debug it then!');
    return
end

```

REFERENCES

- [1] E. L. Allgower, D. J. Bates, A. J. Sommese, and C. W. Wampler, "Solution of polynomial systems derived from differential equations," *Comp.*, vol. 76, no. 1-2, pp. 1–10, Jan. 2006.
- [2] A. J. Sommese and C. W. Wampler, *The Numerical Solution of Systems of Polynomials. Arising in Engineering and Science.* World Scientific Publishing Co. Pte. Ltd, 2005.
- [3] J. W. Thomas, *Numerical partial differential equations. Conservation laws and elliptic equations.* Springer-Verlag, New York, 1999, vol. 2.
- [4] K. Georg, "A new exclusion test," *J. Comp. Appl. Math.*, vol. 152, pp. 147–160, March 2003.
- [5] B. Breuer, P. J. McKenna, and M. Plum, "Multiple solutions for a semilinear boundary value problem: a computational multiplicity proof," *J. Diff. Eq.*, vol. 195, pp. 243–269, Nov. 20, 2003.
- [6] B. L. van der Waerden, *Algebra. Vol. I. Based in part on lectures by E. Artin and E. Noether. Translated from the seventh German edition by Fred Blum and John R. Schulenberger.* Springer-Verlag, New York, 1991.
- [7] C. B. Garcia and T. Y. Li, "On the number of solutions to polynomial systems of equations," *SIAM J. Numer. Anal.*, vol. 17, no. 4, pp. 540–546, Aug. 1980.
- [8] B. L. van der Waerden, *Algebra. Vol. II. Based in part on lectures by E. Artin and E. Noether. Translated from the fifth German edition by John R. Schulenberger.* Springer-Verlag, New York, 1991.

- [9] E. L. Allgower, "A survey of homotopy methods for smooth mappings," *Numer. sol. of nonlin. eqs.*, vol. 878 of Lecture Notes in Math., pp. 1–29, 1981.
- [10] K. Georg, "Improving the efficiency of exclusion algorithms," *Adv. Geom.*, vol. 1, pp. 193–210, June 2001.
- [11] E. L. Allgower and K. Georg, *Numerical Continuation Methods: An Introduction*. Springer-Verlag New York Berlin Heidelberg, 1990, vol. 13 of Series in Computational Mathematics.
- [12] T.-Y. Li, "Numerical solution of multivariate polynomial systems by homotopy continuation methods," *Acta numerica*, vol. 6, pp. 399–436, 1997.
- [13] T. Y. Li, "Numerical solution of polynomial systems by homotopy continuation methods," *Handbook of numerical analysis*, vol. XI, pp. 209–304, 2003.
- [14] H. B. Keller, *Lectures on numerical methods in bifurcation problems*. Springer-Verlag, Berlin, 1987, with notes by A. K. Nandakumaran and Mythily Ramaswamy. Tata Institute of Fundamental Research Lectures on Mathematics and Physics, 79. Published for the Tata Institute of Fundamental Research, Bombay.
- [15] M. H. Protter and H. F. Weinberger, *Maximum Principles in Differential Equations*. Springer-Verlag, New York, 1984.
- [16] T. Laetsch, "The number of solutions of a nonlinear two point boundary value problem," *Indiana Univ. Math. J.*, vol. 20, pp. 1–13, 1970/1971.
- [17] H. B. Keller, *Numerical methods for two-point boundary value problems*. Dover Publications, Inc., New York, 1992, corrected reprint of the 1968 edition.
- [18] B. Gidas, W.-M. Ni, and L. Nirenberg, "Symmetry and related properties via the maximum principle," *Comm. Math. Phys.*, vol. 68, pp. 209–243, 1979.

- [19] R. E. Moore, "A test for existence of solutions to nonlinear systems," *SIAM J. Numer. Anal.*, vol. 14, no. 4, pp. 611–615, Sept. 1977.
- [20] J. S. Zhang, W. Wang, and B. L. Chen, "A cell exclusion algorithm for finding all global minimizers of functions of several variables," *Math. Numer. Sinica*, vol. 17, pp. 443–455, 1995.
- [21] Z.-B. Xu, J.-S. Zhang, and W. Wang, "A cell exclusion algorithm for determining all the solutions of a nonlinear system of equations," *Appl. Math. Comput.*, vol. 80, pp. 181–208, Dec. 1996.
- [22] Z.-B. Xu, J.-S. Zhang, and Y.-W. Leung, "A general cdc formulation for specializing the cell exclusion algorithms of finding all zeros of vector functions," *Appl. Math. Comput.*, vol. 86, pp. 235–259, Oct. 1997.
- [23] B. L. Meng and Z. D. Xing, "A cell exclusion algorithm for finding all solutions to systems of nonlinear equations," *Pure Appl. Math. (Xi'an)*, vol. 15, pp. 72–76, 1999.
- [24] E. Allgower, M. Erdmann, and K. Georg, "On the complexity of exclusion algorithms for optimization," *J. Complex.*, vol. 18, pp. 573–588, June 2002.
- [25] M. I. Syam, "Nonlinear optimization exclusion tests for finding all solutions of nonlinear equations," *Appl. Math. Comput.*, vol. 170, pp. 1104–1116, Nov. 2005.
- [26] R. Precup, *Methods in Nonlinear Integral Equations*. Kluwer Academic Publishers, 2002.
- [27] Y. S. Choi and P. J. McKenna, "A mountain pass method for the numerical solution of semilinear elliptic problems," *Nonlin. Anal.*, vol. 20, pp. 417–437, Feb. 1993.
- [28] G. Chen, J. Zhou, and W.-M. Ni, "Algorithms and visualization for solutions of nonlinear elliptic equations," *Internat. J. Bifur. Chaos Appl. Sci. Eng.*, vol. 10, pp. 1565–1612, July 2000.

- [29] G. Chen, W.-M. Ni, A. Perronnet, and J. Zhou, "Algorithms and visualization for solutions of nonlinear elliptic equations. ii. dirichlet, neumann and robin boundary conditions and problems in 3d." *Internat. J. Bifur. Chaos Appl. Sci. Eng.*, vol. 11, pp. 1781–1799, July 2001.
- [30] Y. S. Choi, P. J. McKenna, and M. Romano, "A mountain pass method for the numerical solution of semilinear wave equations," *Numer. Math.*, vol. 64, no. 1, pp. 487–509, Dec. 1993.
- [31] P. H. Rabinowitz, "Minimax methods in critical point theory with applications to differential equations," *CBMS Regional Conf. Ser. in Math.*, no. 65, 1986.
- [32] A. Ambrosetti and G. Prodi, "On the inversion of some differentiable mappings with singularities between banach spaces," *Ann. Math. Pura Appl.*, vol. 93, pp. 231–246, 1972.
- [33] A. Lazer and P. McKenna, "On the number of solutions of a nonlinear dirichlet problem," *J. Math. Anal. Appl.*, vol. 84, pp. 282–294, Nov. 1981.
- [34] E. L. Allgower and K. Georg, *Numerical Path Following*. Noth Holland, 1997, vol. V of Handbook of Numerical Analysis.
- [35] C. J. Budd, A. R. Humphries, and A. J. Wathen, "The finite element approximation of semilinear elliptic partial differential equations with critical exponents in the cube," *SIAM J. Sci. Comput.*, vol. 20, no. 5, pp. 1875–1904, 1999.