Bifurcation analysis of the dynamical system with numerical continuation

A dissertation submitted for the degree of

Master of Science

Of

University College London

Xinyu Yao 21052295

Prof. Edward Johnson, Supervisor

DATE 2022.8

Abstract

The dynamical system which describes the changes of the state quantity (a point or vector in the appropriate space) concerning time exists everywhere in real life. Currently, research related to it has become a new scientific trend but is still not very complete. This project will combine numerical time integration approaches (finite-difference and finite-element methods) and numerical continuation schemes to study the bifurcation analysis of some dynamical system problems which may depend on parameters with particular meanings. A detailed mathematical explanation related to these methods will be covered in this dissertation. Finally, we apply such methodology to the Swift-Hohenberg equation to make an analysis of the bifurcation and solution diagrams obtained by our research program.

Keywords: dynamical system, bifurcation analysis, finite-difference, finite-element, continuation

Link of the GitHub repository: `https://github.com/UCLyxy/PHAS0077_Xinyu`

Acknowledgement

Initially, I would like to express my deep gratitude to my research supervisor Prof. Edward Johnson. He gave me constructive guidance and explanations during weekly meetings and emails. Each time, he listens to my presentation carefully and points out my mistakes as soon as possible, making me deepen my comprehension of the research project and become more interested in this topic.

Secondly, my great thanks are also extended to Dr. Sergei N Timoshin and Prof. Gert van der Heijden. They taught me finite-difference methods and finite-element methods in module MATH0086, helping me a lot with my research.

Finally, I would like to thank all the lecturers in Scientific Computing. They helped me learn a lot of professional knowledge, especially in numerical analysis.

Contents

Chapter 1

Introduction

1.1 Background about Dynamical System

The dynamical system which describes the changes of the **state quantity** (a point or vector in the appropriate space) with respect to the **time** exists everywhere in real life. Research related to it has become a new scientific trend but still not very complete. In mathematics, most of the dynamical system could be approximated by the **differential equations or difference equations** very closely.

For example (Biology), consider a population of y animals in an environment where a maximum of B individuals can coexist. Assume that the initial population y_0 satisfies that $y_0 < B$ and that for small populations $y << B$ the reproduction rate of the animals is approximately equal to constant C. For larger populations the rate should slow down so that the population never exceeds B. The simplest model for such a situation is the logistic equation:

$$\begin{aligned} &\frac{dy}{dt} = Cy(t)(1 - \frac{y(t)}{B}) \\ &y(0) = y_0 \end{aligned} \tag{1.1}$$

Through solving this equation we could get the information about the evolution of the population with respect to time.

Moreover, this differential equation could be extended to simulate two interacting populations in competition for the same resources. Assume that a predator with population y_2 survives by hunting a prey with population y_1. Then the change of the population of these two interacting species could be modeled by the following system of differential equations [9](Lotka-Volterra predator-prey model):

$$\begin{aligned} \frac{dy_1}{dt} &= C_1 y_1(t)[1 - b_1 y_1(t) - d_2 y_2(t)] \\ \frac{dy_2}{dt} &= -C_2 y_2(t)[1 - b_2 y_2(t) - d_1 y_1(t)] \end{aligned} \tag{1.2}$$

where C_1 and C_2 are the growth factors of the two populations, d_1 and d_2 accounts for interaction between the population(for example, preying or contamination), and b_1 and b_2 are related to the amount of resources such like food available for each population.

Unfortunately, the exact solution of such an equation is nearly impossible to get in usual.

1.2 Research Method

A widely-applicable approach [8] to make some scientific research related to the dynamic system is just to find out some objects with **compact invariant properties** such as **equilibrium, limit cycles(periodic orbits) or even invariant sets**. Then to observe the flow of the trajectory of this dynamic system, we just need to study the **local behaviour** around these compact invariant objects. In this case, the purpose is just to compute the qualitative(or even quantitative if possible) solutions of these local behaviours that may depend on some key parameters(local behaviour might change with respect to these varied parameters). Such method in dynamical system is also called **stability analysis** [11] or **bifurcation analysis**. Normally, the main result of this step would be manifested by a **bifurcation diagram**, which represents all the information on the division of parameter space into regions of topologically different behaviour as well as the phase portraits.

However, even though some models may look trivial, it is extremely difficult to implement theoretical ideas without **numerical methods**. Most commonly, numerical time integration (such as **Runge-Kutta methods**) would be applied to such case by solving the initial value problems. However, the numerical continuation (especially **pseudo arclength continuation**) would be much more suitable when it comes to the research of how the local behaviour changes as a function of parameters.

Hence this project would combine both numerical methods (finite-difference and finite element scheme) as well as continuation ideas to make bifurcation analysis for a dynamical system.

1.3 Research Aims and Objectives

The purpose of this project is to provide a feasible approach (in practice, a Matlab continuation program) for users to explore the trajectory flow, and compare solution variation caused by bifurcation of the dynamical systems which could be generated by differential equations in the form of [10] :

$$M\partial_t \boldsymbol{u} = \nabla \cdot (c \times \nabla \boldsymbol{u}) - a\boldsymbol{u} + b \times \nabla \boldsymbol{u} + f \tag{1.3}$$

where $\boldsymbol{u} = \boldsymbol{u}(x,t) \in \mathbb{R}^n$, $t \geq 0$, $x \in I$ which is an interval (we only focus our attention on 1D dimension problem in this project). The coefficients c, a, b, f may denpend on x (equation variable) or $\lambda \in \mathbb{R}$ (equation parameter). $M \in \mathbb{R}^{n\times n}$ is a mass matrix (generated from tansforming a higher order differential equation into a differential equation **system**).

In order to achieve this purpose and make users understand the basic theoretical idea (mathematical principle) behind the program, this dissertation would review both super-critical numerical schemes for solving differential equations (finite-difference and finite-element) and give a detailed explanation of continuation thoughts. The specific dissertation structure would be presented in the next section.

1.4 Dissertation Structure

Following the background and introduction, Chapter 2 will review finite-difference and finite-element methods, as well as their concrete example application discussions (Allen-Cahn equation [4] for finite-element scheme and Brusselator model [15] for finite-difference method).

Chapter 3 will focus on the explanation of the mathematical background related to the continuation thoughts, including basic continuation ideas (natural continuation and arclength continuation), bifurcation test, and switching branch algorithm, which consists of the main methodology of the program. Moreover, a detailed methodology implementation of the Swift-Hohenberg equation will be covered at the tail.

Chapter 4 will mainly present and describe some bifurcation and solution diagrams of the Swift-Hohenberg equation obtained by our methodology and program.

Finally, Chapter 5 concludes this project with a brief evaluation including a summary of the study's implications, limitation, and recommendation.

Chapter 2

Literature Review

2.1 Introdunction

In the real-world, most problems in physics or engineering are just described by ordinary differential equations or partial differential equations. However, in most cases, the analytical solution or exact solution is extremely difficult (almost impossible) to get. Therefore, the existence of numerical methods does make sense at this point.

This chapter will present two main numerical methods that widely applied in solving differential equations. One is the finite difference method, the other one is the finite element scheme.

The Finite-Element scheme is one of the most classic numerical methods to get the approximation of solutions to differential equations, especially when the geometry of boundary conditions (BCs) is quite complicated. This method would be reviewed in the next section, including its general procedure as well as one concrete example(the Allen-Cahn equation).

When it comes to the finite difference method, instead of approximating solutions such as finite element, this method normally approximates the derivative elements in the equation by **finite differences**, which would be discussed concretely in the last section of this chapter.

2.2 Finite Element Scheme

2.2.1 General Procedure of Finite Element Scheme

- **Separation**: separate the domain of differential equations into some small pieces (which are defined as **the finite elements** [14]). To make the method more feasible, the geometry of these **finite elements** should be chosen carefully(as simple as possible). For instance, triangle or rectangle.
- **Discretisation**: select some nodes on the **finite elements** where the approximate value would be calculated by the **Weighed Residual method** , which would be introduced in next section
- **Interpolation** Once nodal values are approached, appropriate interpolation methods should be applied between these nodal values get the solutions on the entire domain

2.2.2 Advantages of Finite Element Scheme

- general, strong, robust(any boundary value problems could be solved by **Finite Element method**
- it is convenient for **Finite Element method** to refine the structure to get different required accuracy
- Instead of **Finite Difference method**, **Finite Element method** approximate solutions of differential equations rather than differential equation itself at some particular point

2.2.3 Weighted Residual Method

Consider the boundary-value problem:

$$\frac{d^2u}{dx^2} + u + x = 0 \quad x \in [0,1]$$
$$u(0) = u(1) = 0$$

Suppose that a trial function(initial guess) $\tilde{u}$ at least satisfies the Dirichlet boundary conditions such that:

$$\tilde{u} = ax(1-x)$$

Then the residual denoted as $R(x)$ of the equation would be:

$$R(x) = \frac{d^2\tilde{u}}{dx^2} + \tilde{u} + x$$

It is possible for us to adjust the value of a to get the "best" approximation (minimum residual). However, how could be find or define the "best"? The Weighted Residual scheme defines a wight function ω and requires the weighted average to be 0 such that:

$$I := \int_0^1 \omega R dx = 0$$

Remark: this gives an equation for the single unknown variable a. Hence given ω, we can solve for a.
However, how to choose the weight function ω?

Galerkin method

For ω take the trial function itself:

$$\omega(x) = x(1-x)$$

Then

$$0 = I = \int_0^1 x(1-x)[-2a + ax(1-x) + x]dx$$

which gives a to be $\frac{8}{15}$

Improvement of approximation

To improve the approximation we can take two terms in the trial solution(in this case, $\tilde{u}$ still satisfies the boundary conditions):

$$\tilde{u} = a_1 x(1-x) + a_2 x^2(1-x)$$

Since there are two unknown variables a_1 and a_2, we need two equations.

Galerkin Method

Here we have two trial functions $x(1-x)$ and $x^2(1-x)$. By Galerkin method, we let ω to be trial function itself such that:

$$\omega_1(x) = x(1-x)$$
$$\omega_2(x) = x^2(1-x)$$

Thus we get two equations:

$$\int_0^1 R\omega_1 dx = 0$$
$$\int_0^1 R\omega_2 dx = 0$$

Remark Since $\tilde{u}$ could be any function satisfies the boundary conditions, more generally, we could use n term trial function:

$$\tilde{u} = \sum_{i=1}^{n} a_i x^i (1-x)$$

or Fourier series:

$$\tilde{u} = \sum_{i=1}^{n} a_i sin(n\pi x)$$

General Form

Assume the differential equation is in the form of $Lu = f$ where L is a linear differential operator. We use the n term trial function

$$\tilde{u} = \sum_{i=1}^{n} a_i \varphi_i$$

to approximate the solution, where φ_i are basis trial functions.
The residual is:

$$R = L\tilde{u} - f = L\sum_{i=1}^{n} a_i \varphi_i - f$$

By linearity:

$$R = \sum_{i=1}^{n} a_i L\varphi_i - f$$

We use Galerkin method to get n equations such that:

$$\begin{aligned} &\omega_j = \varphi_j \\ &0 = \int_a^b \varphi_j R dx = \sum_{i}^{n} a_i \int_a^b \varphi_j L\varphi_i dx - \int_a^b f\varphi_j dx \end{aligned}$$

where $j = 1, 2, \ldots, n$

2.2.4 Weak Formulation

Consider again the boundary-value problem:

$$\begin{aligned} &\frac{d^2u}{dx^2} + u + x = 0 \quad x \in [0,1] \\ &u(0) = u(1) = 0 \end{aligned}$$

The existence of derivative term $\frac{d^2u}{dx^2}$ is so-called **strong formulation** [16]. It is fine for the analytical solution to be twice differentiable. However, what we are interested in is the numerical solution, which served as an approximation. In this case, the existence of a second derivative is overly restrictive. For example, for picewise linear trial solution $\tilde{u}$, $\frac{d^2u}{dx^2}$ is not defined, hence weighted residual method would fail. However, this issue could be solved by using a technique called weak formulation obtained as follows:

Multiply the differential equation by a **test function** v which is smooth enough and integrate:

$$\int_0^1 (\frac{d^2u}{dx^2} + u + x) v dx = 0$$

By applying integration by part:

$$-\int_0^1 (\frac{du}{dx}\frac{dv}{dx} - uv - xv)dx + [\frac{du}{dx}v]_0^1 = 0$$

Provided that u solves weak formulation for all possible test functions v, then we could expect u also solves the stong formulation.

2.2.5 Example Analysis

Allen-Cahn equation with homogeneous Neumann boundaries

$$\partial_t u = c\Delta u + \lambda u + u^3 - \gamma u^5 \tag{2.1}$$

where $u = u(x,t) \in \mathbb{R}$, $t \geq 0$, $x \in [a,b]$, $\lambda \in \mathbb{R}$ is a parameter
all the other coefficients [10] c, γ may depend on x, λ and, typically nonlinearly, on u or $f(u)$, where f may be a nonlinear function.

To make the problem a little bit easier, here we choose a homogeneous Neumann boundaries conditions, that is
Boundary conditions:

$$\partial_x u|_{x=a} = 0 \quad \partial_x u|_{x=b} = 0 \tag{2.2}$$

Next, to make a study on this dynamical system, here we should focus on the steady state problem (the time derivative of u should equal to 0), that is:

$$G(u,\lambda) := c\Delta u + \lambda u + u^3 - \gamma u^5 = 0 \tag{2.3}$$

Finite Element scheme

Separation and Discretisation
An 1D element is simply an interval. On a single element(one interval), we just label as many nodes $u_i, u_j, \ldots$ as we want to denote the unknown field variable [17] (for instance, the displacement at one particular position of an interval).
The core of Finite Element scheme is to approximate the solution, hence we use the trial function:

$$\hat{u} = \sum_{j=1}^{n_p} \phi_j(x) u_j \tag{2.4}$$

where n_p is the number of nodes on the entire domain, u_j are the nodal values(whatever they are at this moment, they will be fixed(approximated) by using the equations), ϕ_j are the shape functions, which determine the interpolation between these nodal values.
Denote $\boldsymbol{u} \in \mathbb{R}^{n_p}$ as the vector which collects all the nodal values.

weak formulation

Since our aim is to find the approximation of the exact solutions, the existence of a second derivative is overly restrictive.

Multiple the equation (2.3) by an arbitrary function $v \in C^1$ and integrate over the domain:

$$0 = \int_a^b (c\frac{d^2u}{dx^2} + \lambda u + u^3 - \gamma u^5)v dx \tag{2.5}$$

Lemma 1 (Green's second identity)**.**

$$\iint_\Omega \phi \frac{\partial \psi}{\partial x} dxdy = -\iint_\Omega \frac{\partial \phi}{\partial x} \psi dxdy + \oint_{\partial\Omega} \phi\psi n_x ds \tag{2.6}$$

By using Green's second identity (integration by part in 1D dimension):

$$\begin{aligned} 0 &= \int_a^b (c\frac{d^2u}{dx^2} + \lambda u + u^3 - \gamma u^5)v dx \\ &= -c\int_a^b \frac{du}{dx}\frac{dv}{dx}dx + [c\frac{du}{dx}v]|_a^b + \int_a^b (\lambda u + u^3 - \gamma u^5)v dx \\ &= -c\int_a^b \frac{du}{dx}\frac{dv}{dx}dx + [c\frac{du}{dx}v]|_a^b + \int_a^b f(u)v dx \end{aligned} \tag{2.7}$$

where $f(u) = \lambda u + u^3 - \gamma u^5$
By consider the homogeneous Neumann boundary conditions, we get the weak formulation of equation (2.3)

$$0 = -c\int_a^b \frac{du}{dx}\frac{dv}{dx}dx + \int_a^b f(u)v dx \tag{2.8}$$

It is clear to see that in this weak formulation, the existence of a first derivative is enough.

Finite Element equation

Substituting the trial function (2.4) into weak formulation:

$$c\int_a^b \sum_{j=1}^{n_p} \frac{d\phi_j}{dx}\frac{dv}{dx}dx u_j = \int_a^b f(\sum_{j=1}^{n_p} \phi_j u_j)v dx \tag{2.9}$$

Here we have n_p unknowns, in this case, we need n_p equations. By using Galerkin's method, let $v_i = \phi_i$, the left hand side of equation (2.9) would become:

$$c\int_a^b \sum_{j=1}^{n_p} \frac{d\phi_i}{dx}\frac{d\phi_j}{dx}dx u_j \tag{2.10}$$

At the same time, we use

$$\int_a^b \sum_{j=1}^{n_p} \phi_i\phi_j f(u_j)dx$$

to approximate [10] the right hand side of equation (2.9) and thus getting the **Finite Element equation**:

$$c\int_a^b \sum_{j=1}^{n_p} \frac{d\phi_i}{dx}\frac{d\phi_j}{dx} dx u_j - \int_a^b \sum_{j=1}^{n_p} \phi_i\phi_j f(u_j) dx = 0 \tag{2.11}$$

Let

$$K_{ij} = c\int_a^b \frac{d\phi_i}{dx}\frac{d\phi_j}{dx} dx$$

$$M_{ij} = \int_a^b \phi_i\phi_j dx$$

where K is also called stiffness matrix and M is called mass matrix
Hence the final version of the **Finite Element equation** (i-th equation) would be :

$$K_{ij}u_j - M_{ij}f(u_j) = 0 \tag{2.12}$$

In fact, equation (2.12) is the Finite Element approximation of equation (2.3). Numerically, the form of equation (2.12) is much easier to compute, hence this approximation does make sense.

2.3 Finite Difference Method

2.3.1 Problem Setting

Consider inital value problem of the following form: $f : \mathbb{R} \times [t_0, \infty] \to \mathbb{R}$, continuously differentiable with respect to both arguments, and initial data $y_0 \in \mathbb{R}$ is given, find y satisfying:

$$\begin{cases} \dfrac{dy}{dt} = f(y(t), t) \\ y(t_0) = y_0 \end{cases} \tag{2.13}$$

Problem (2.13) is often called **Cauchy problem** for the differential equation

$$\frac{dy}{dt} = f(y(t), t)$$

The existence and uniqueness of the solution of problem (2.13) could be ensured by the famous fundamental theorem of differential equations, which is also called **Picard's theorem** [13]

Theorem 1 (Picard's theorem)**.** *Suppose that $f : \Omega \to \mathbb{R}$ is continuous on the rectangle $\Omega = [y_0 - c, y_0 + c] \times [t_0, t_{\max}]$, for some $t_{\max} > 0$ and $c > 0$, and that f satisfies a Lipschitz condition in Ω with respect to y, i.e. $\exists L > 0$ such that*

$$|f(v,t) - f(w,t)| \leq L|v - w|, \quad \forall v, w \in [y_0 - c, y_0 + c], \forall t \in [t_0, t_{\max}] \tag{2.14}$$

Suppose also that

$$K := \max_{t \in [t_0, t_{\max}]} |f(y_0, t)| \leq \frac{cL}{e^{Lt_{\max}} - 1} \tag{2.15}$$

Then there exists a unique continuously differentiable function $y : [t_0, t_{\max}] \to [y_0 - c, y_0 + c]$, such that $y(t_0) = y_0$ and $\dfrac{dy}{dt} = f(y(t), t)$ for $t \in [t_0, t_{\max}]$

Remark
The extra condition

$$K := \max_{t \in [t_0, t_{\max}]} |f(y_0, t)| \leq \frac{cL}{e^{Lt_{\max}} - 1}$$

allows us to prove existence and uniqueness of the solution on the specified interval $[t_0, t_{\max}]$ rather than just on some interval $[0, \delta]$, as in more standard versions.

example 1

$$\begin{cases} \dfrac{dy}{dt} = 3y(t) - 3t \\ y(0) = 1 \end{cases} \tag{2.16}$$

Here $f(v,t) = 3v - 3t$ and the exact solution is $y(t) = \frac{2}{3}e^{3t} + t + \frac{1}{3}$.
In this example, $|f(v,t) - f(w,t)| = |(3v - 3t) - (3w - 3t)| = 3|v - w|$ for any $t, v, w \in \mathbb{R}$. Hence the Lipschitz condition (2.14) is satisfied with $L = 3$ and $c > 0$ arbitrary. Given any $t_{\max} > 0$, there exists $K = \max\{3, 3t_{\max}\} = 3\max\{1, t_{\max} - 1\}$, which means that condition (2.15) could be satisfied by taking $c > \max\{1, t_{\max} - 1\}(e^{3t_{\max} - 1})$. Thus the initial value problem (2.16) has a unique global solution for all $t \geq 0$.

example 2

$$\begin{cases} \dfrac{dy}{dt} = \sqrt[3]{y(t)} \\ y(0) = 0 \end{cases} \tag{2.17}$$

Here $f(v,t) = \sqrt[3]{v}$. This problem admits the following three solutions:

$$y(t) = 0, y(t) = \sqrt{\frac{8t^3}{27}}, y(t) = -\sqrt{\frac{8t^3}{27}}$$

However, the lack of unique solvability of this example dose not contract Picard's theorem because the theorem dose not apply in this case. Here $f(y,t)$ is not Lipschitz continuous with respect to y in any neighbourhood of $(0,0)$. In particular, if we choose $w = 0$ in condition (2.14), we get:

$$|f(v,t) - f(w,t)| = |f(v,t)| = \sqrt[3]{v} \tag{2.18}$$

which tends to 0 slower than Lv as v gose to 0, for any $L \in \mathbb{R}^+$, which means that we cannot find the Lipschitz constant such that condition (2.14) satisfied, no matter how small $t_{\max}$ and c are.

example 3

$$\begin{cases} \dfrac{dy}{dt} = 1 + y(t)^2 \\ y(0) = 0 \end{cases} \tag{2.19}$$

Here we have an exact solution $y(t) = \tan(t)$ in the interval $(0, \dfrac{\pi}{2})$. However, it is clear to see that $y(t) \to \infty$ as $t \to \dfrac{\pi}{2}$, hence the solution is only defined locally rather on the whole interval $(0, \infty)$. At the same time, we observe that:

$$|f(v,t) - f(w,t)| = |v^2 - w^2| = |v + w||v - w| \quad \forall t, v, w \in \mathbb{R} \tag{2.20}$$

since $\forall v, w \in (-c, c)$, the Lipschitz condition (2.14) holds with Lipschitz constant $L = 2c$. In addition, $|f(0,t)| = 1$. In this case, with condition $2c^2 \geq e^{2ct_{\max}} - 1$, (2.15) could be satisfied for $t_{\max}$ sufficiently small by selecting c carefully, hence local existence and uniqueness of solution could be achieved. However, when $t_{\max}$ is large enough(for instance, $t_{\max} > 2$ is sufficiently large), this inequality (2.15) cannot be achieved by any $c > 0$. As a result, we cannot use Picard's theorem to get gloabl existence and uniqueness, which is consistent with the observation that the exact solution blows up at finite t.

Remark

The Picard's theorem breaks down before the exact solution actually blows up at $t = \dfrac{\pi}{2}$

2.3.2 Numerical Discretization

From now on, we could adopt the finite difference method to obtain an approximate solution to the initial value problem (2.13). First of all, start by selecting an positive integer $N \geq 1$ and discretizing the interval $[t_0, t_{\max}]$ into N pieces subintervals $[t_n, t_{n+1}]$ where the mesh points (not necessarily uniformly distributed) satisfy:

$$t_0 < t_1 < t_2 < \cdots < t_n < t_{n+1} < \cdots < t_N = t_{\max}$$

and the corresponding step size is defined by:

$$h_n := t_{n+1} - t_n \quad n = 0, 1, 2, \ldots, N-1$$

The numerical approximation of the continuously differentiable function $y(t)$ will consist of a sequence of $N+1$ elements, which could be denoted as: $\{y_n\}_{n=0}^{N}$. For each integer n, the element y_n represents an approximation of the exact value of $y(t_n)$.

Then what we need to do is just to apply relevant finite difference scheme to derive a model (usually a difference equation for iteration) satisfied by $\{y_n\}_{n=0}^{N}$, which means it is necessary for us to replace the original differential equation $\frac{dy}{dt} = f(y(t), t)$ by a suitable approximation. Here we start by the most simple case.

2.3.3 Forward Euler Scheme

Recall the original definition of derivative:

$$\frac{dy}{dt} = \lim_{h \to 0} \frac{y(t+h) - y(t)}{h}$$

Hence one approach to approximate the derivative at point $t = t_n$ is using a difference quotient:

$$\frac{y(t_n + h) - y(t_n)}{h}$$

for some finite step size $h > 0$. Here a normal and simple choice is just to take $h_n = h$ (make numerical discretization uniformly distributed). Finally, replaceing function $f(y(t), t)$ by $f(y_n, t_n)$ leads to the numerical method for solving problem (2.13), the **Forward(explicit) Euler method**:

$$y_{n+1} = y_n + hf(y_n, t_n), \quad n = 0, 1, \ldots, N-1 \tag{2.21}$$

This method is explicit since we could get y_{n+1} without solving an equation(the right hand side of (2.21) is independent of y_{n+1} and hence y_{n+1} could be calculated in a straightforward way). However, the price that we have to pay for this convenience is the stability.

error analysis

At the grid point t_{n+1}, we have $y(t_{n+1}) = y(t_n + h)$. Assume $y(t)$ is sufficiently smooth (i.e. $y \in C^\infty$), we could get a Taylor expansion:

$$y(t_{n+1}) = y(t_n + h) = y(t_n) + h\frac{dy}{dt}|_{t=t_n} + O(h^2) \tag{2.22}$$

with the error written as an order-of-magnitude term. By our differential equation $\frac{dy}{dt} = f(y(t), t)$, it is equivalent to

$$y(t_{n+1}) = y(t_n + h) = y(t_n) + hf(y(t_n), t_n) + O(h^2) \tag{2.23}$$

and hence we get:

$$\frac{y(t_{n+1}) - y(t_n)}{h} = f(y(t_n), t_n) + O(h^2) \tag{2.24}$$

By equation (2.23) and (2.24), we get the result that **Forward Euler method** is first-order accurate in the equation and second-order accurate in the solution(since the local truncation error in the solution satisfy $y_{n+1} - y(t_{n+1}) = O(h^2)$).

In this case, it is clearly to see that the accuracy of **Forward Euler method** is not good enough(if we would like to make our numerical approximation as accurate as possible, we have to reduce step size h significantly, leading to a higher computation price). Hence we have to introduce a better finite difference scheme (with higher order of accuracy) in this sense.

2.3.4 Taylor Series Method

Although of little practical significance, Taylor series methods offer a critical starting point for the Runge-Kutta algorithm. We still hold this assumption that $y(t)$ is sufficiently smooth(all necessary partial derivatives exist), the $p-th$ order Taylor expansion of $y(t)$ at point $t = t_{n+1} = t_n + h$ is:

$$\begin{aligned} y(t_n + h) =& y(t_n) + h\frac{dy}{dt}|_{t=t_n} + \frac{h^2}{2}\frac{d^2y}{dt^2}|_{t=t_n} + \dots \\ &+ \frac{h^p}{p!}\frac{d^py}{dt^p}|_{t=t_n} + \frac{h^{p+1}}{(p+1)!}\frac{d^{p+1}y}{dt^{p+1}}|_{t=\xi} \end{aligned} \tag{2.25}$$

For some ξ between t_n and $t_{n+1} = t_n + h$

From our differential equation $\frac{dy}{dt} = f(y(t), t)$, we have:

$$\begin{aligned} \frac{dy}{dt}|_{t=t_n} &= f(y(t_n), t_n) \\ \frac{d^2y}{dt^2}|_{t=t_n} &= f'(y(t_n), t_n) \\ &\dots \\ \frac{d^py}{dt^p}|_{t=t_n} &= f^{p-1}(y(t_n), t_n) \end{aligned} \tag{2.26}$$

Here dash is used to denote the **total derivative** with respect to t, for instance:

$$f'(y(t),t) = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}\frac{dy}{dt} = \frac{\partial f}{\partial t} + f\frac{\partial f}{\partial y}$$

By truncating the remainder term of (2.25) and replacing the analytical values $y(t_n)$ with approximation y_n, a new finite difference scheme could be obtained:

$$y_{n+1} = y_n + hf(y_n,t_n) + \frac{h^2}{2}f'(y_n,t_n) + \cdots + \frac{h^p}{p!}f^{p-1}f(y_n,t_n) \tag{2.27}$$

It is clearly to see that the local truncation error of Taylor series method is $p - th$ order accurate in the equation and $p + 1 - th$ order accurate in the solution. With this method, we could get a numerical method as accurate as we want!
Remark
When p=1, this method reduce to **Forward Euler scheme**

However, Taylor series method has a fatal flaw–computation cost, especially when the value of p is large(many partial derivatives have to be computed at each iteration). In this case, we need a numerical method which maintains the accuracy and avoid the computation of partial derivatives.

2.3.5 Runge-Kutta Methods

The second-order Taylor series method is:

$$\begin{aligned} y_{n+1} =& y_n + hf(y_n,t_n) + \frac{h^2}{2}f'(y_n,t_n) \\ =& y_n + hf(y_n,t_n) + \frac{h^2}{2}\frac{\partial f}{\partial t}|_{t=t_n} + \frac{h^2}{2}(f\frac{\partial f}{\partial y})|_{t=t_n} \end{aligned} \tag{2.28}$$

For the aim of keeping accuracy and not involving computation of two partial derivative terms, we have to search for an approximation of the form:

$$y_{n+1} = y_n + h[\omega_1 f(y_n,t_n) + \omega_2 f(y_n + Y, t_n + T)] \tag{2.29}$$

Our purpose is to reproduce the linear as well as the quadratic terms with respect to h. Hence, perturbation Y and T are chosen to be:

$$\begin{aligned} Y =& \alpha h f(y_n,t_n) \\ T =& \lambda h \end{aligned} \tag{2.30}$$

where α and λ are constants.

Substituting into (2.29) and Taylor expanding with respect to h:

$$\begin{aligned} y_{n+1} &= y_n + h(\omega_1+\omega_2)f(y_n,t_n) \\ &+ h^2\omega_2[\alpha f(y_n,t_n)\frac{\partial f}{\partial y}|_{t=t_n} + \lambda\frac{\partial f}{\partial t}|_{t=t_n}] + O(h^3) \end{aligned} \tag{2.31}$$

Equations (2.28) and (2.29) are identical to $O(h^3)$ if we choose parameters:

$$\begin{aligned} \omega_1+\omega_2 &= 1 \\ \alpha\omega_2 &= 0.5 \\ \lambda\omega_2 &= 0.5 \end{aligned} \tag{2.32}$$

Overall, equation (2.28) could approximate $y(t_n)$ with a local truncation error $O(h^3)$ and at the same time equation (2.29) could approximate (2.28) still with a local truncation error $O(h^3)$. Hence if we use equation (2.29) to approximate $y(t_n)$, the local truncation error in the solution is still at the order of $O(h^3)$ without any computation of partial derivatives!

example

The second-order Runge-Kutta(RK2) algorithm is in the form of:

$$\begin{aligned} y_{n+1} &= y_n + h[a_1 f(y_n,t_n) + a_2 F_n] \\ F_n &= f(y_n + a_3 h f(y_n,t_n), t_n + a_4 h) \end{aligned} \tag{2.33}$$

where a_{1-4} are constant parameters
For appropriate parameter choice, this method would be second-order accurate and has a local truncation error(solution) in the form of: $Mh^3 + O(h^4)$
Proof :

Using Taylor expansion to F_n

$$\begin{aligned} y_{n+1} &= y + h(a_1 f + a_2 F) \\ F &= f + a_3 h f f_y + a_4 h f_t + \frac{1}{2}a_3^2 h^2 f^2 f_{yy} \\ &+ a_3 a_4 h^2 f f_{ty} + \frac{1}{2}a_4^2 h^2 f_{tt} + O(h^4) \end{aligned} \tag{2.34}$$

For neat writing, we omit arguments y_n and t_n. At the same time:

$$y(t_n+h) = y + hy' + \frac{1}{2}h^2 y'' + \frac{1}{6}h^3 y''' + O(h^4) \tag{2.35}$$

By differential equation, we have :

$$\begin{aligned} y' &= f \\ y'' &= f' = f_t + f f_y \\ y''' &= f'' = f_{tt} + 2f f_{ty} + f_t f_y + f^2 f_{yy} + f(f_y)^2 \end{aligned} \tag{2.36}$$

Compare equation (2.34) and (2.35):
At $O(1)$, $y = y$, all terms cancel
At $O(h)$, to ensure this method is consistent with second-order accurate Taylor series method, we require:

$$y' = (a_1 + a_2)f = f$$

hence we should choose $a_1 + a_2 = 1$
At $O(h^2)$, we let

$$\frac{1}{2}y'' = a_2 a_3 f f_y + a_2 a_4 f_t$$

since $y' = f_t + f f_y$, we set $a_2 a_3 = \dfrac{1}{2}$ and $a_2 a_4 = \dfrac{1}{2}$
At $O(h^3)$, the difference(which is the local truncation error of approximation) between two equations is:

$$[\frac{1}{6}y''' - \frac{1}{4}a_3 f^2 f_{yy} - \frac{1}{2}a_4 f f_{yt} - \frac{1}{4}a_4 f_{tt}]h^3 + O(h^4) \tag{2.37}$$

Replacec y''' by term $f_{tt} + 2f f_{ty} + f_t f_y + f^2 f_{yy} + f(f_y)^2$, we get our final form of local truncation error:

$$[(\frac{1}{6} - \frac{a_3}{4})f^2 f_{yy} + \frac{1}{6}f f_y^2 + (\frac{1}{3} - \frac{a_4}{2})f f_{yt} + \frac{1}{6}f_y f_t + (\frac{1}{6} - \frac{a_4}{4})f_{tt}]h^3 + O(h^4) \tag{2.38}$$

which is exact in the form of $Mh^3 + O(h^4)$

2.3.6 Application: Bifurcation analysis of Brusselator model

The Brusselator is a model of certain autocatalytic reactions [15] described in terms of chemical concentrations $x(t)$ and $y(t)$ where t represents time.

The governing equations are:

$$\begin{aligned} \frac{dx}{dt} &= A - Bx + x^2 y - x \\ \frac{dy}{dt} &= Bx - x^2 y \end{aligned} \tag{2.39}$$

where A and B are both positive parameters. We investigate the behaviour of this system by a finite-difference fourth-order accurate Runge-Kutta approximation. Here is the main code for iteration:

```
for i=1:n
        x_vec(i+1)=x_vec(i)+(1./6).*(k1_vec(i)+2.*k2_vec(i)
           +2.*k3_vec(i)+k4_vec(i));
        y_vec(i+1)=y_vec(i)+(1./6).*(l1_vec(i)+2.*l2_vec(i)
           +2.*l3_vec(i)+l4_vec(i));

        k1_vec(i+1)=h.*f(x_vec(i+1),y_vec(i+1));
        l1_vec(i+1)=h.*g(x_vec(i+1),y_vec(i+1));

        k2_vec(i+1)=h.*f(x_vec(i+1)+0.5.*k1_vec(i+1),y_vec(i
            +1)+0.5.*l1_vec(i+1));
        l2_vec(i+1)=h.*g(x_vec(i+1)+0.5.*k1_vec(i+1),y_vec(i
            +1)+0.5.*l1_vec(i+1));

        k3_vec(i+1)=h.*f(x_vec(i+1)+0.5.*k2_vec(i+1),y_vec(i
            +1)+0.5.*l2_vec(i+1));
        l3_vec(i+1)=h.*g(x_vec(i+1)+0.5.*k2_vec(i+1),y_vec(i
            +1)+0.5.*l2_vec(i+1));

        k4_vec(i+1)=h.*f(x_vec(i+1)+k3_vec(i+1),y_vec(i+1)+
            l3_vec(i+1));
        k4_vec(i+1)=h.*g(x_vec(i+1)+k3_vec(i+1),y_vec(i+1)+
            l3_vec(i+1));
end
```

2.3.7 Bifurcation analysis

Theoretically, the most common way to analyse the trajectory flow of a dynamical system is just to analyse the stability of the equilibrium in this system, and in this Brusselator case, it is clear to see that this system has an equilibrium point

$$x^\star = A, y^\star = \frac{B}{A} \tag{2.40}$$

Then we compute the Jacobian Matrix at this equilibrium:

$$\begin{pmatrix} B-1 & A^2 \\ -B & -A^2 \end{pmatrix} \tag{2.41}$$

Hence the characteristic equation for calcalation of eigenvalues is:

$$\lambda^2 + (A^2 - B + 1)\lambda + A^2 = 0 \tag{2.42}$$

this equilibrium is stable if and only if the real part of both eigenvalues is negative. Since $\lambda_1\lambda_2 = A^2 > 0$(which means that λ_1 and λ_2 have the same sign), $\lambda_1 + \lambda_2$ must be negative, that is, $\lambda_1 + \lambda_2 = B - A^2 - 1 < 0$. Hence the final condition for the stability of this equilibrium is:

$$B < A^2 + 1 \tag{2.43}$$

Next we use RK4 scheme to verify this theoretical result

2.3.8 Numerical Result

stable equilibrium case

Firstly, take $A = 1.0654$ and $B = 2.1051$(which is the case that $B < A^2 + 1$) with 3 different initial conditions. Here is the result:

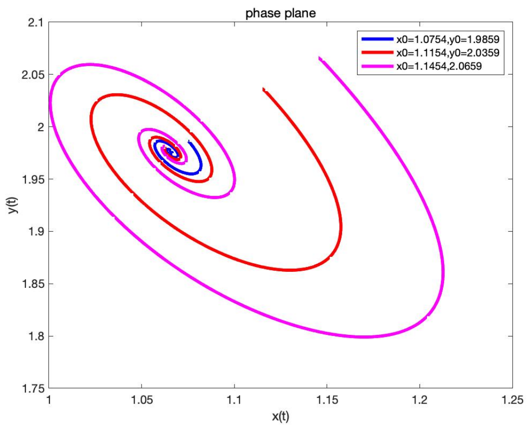


Figure 2.1: oscillatingly stable equilibrium case

which could also be classified as oscillatingly stable(no matter what initial condition we choose, the trajectory curve would converge to this equilibrium oscillatingly with the flow of time). In this case, the eigenvalues λ_1 and λ_2 computed by equation (2.42) are two complex numbers with negative real parts.
Moreover, if we choose $B << A^2 + 1$(for instance, $A = 6, B = 8$), then :

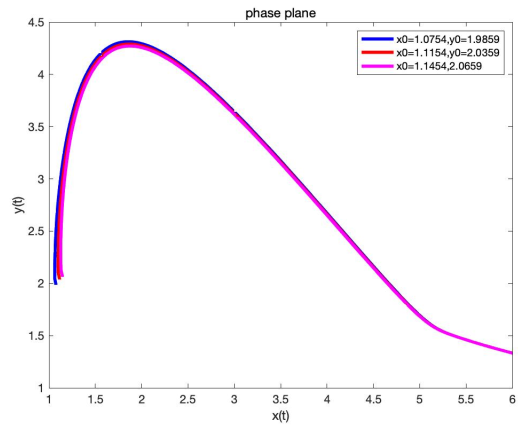


Figure 2.2: monotonically stable equilibrium case

where we could see that as t goes to infinity, all solutions approach the equilibrium

$x^\star = 6, y^\star = 8/6$ monotonically. In this case, we denote the stability type of this equilibrium as monotonically stable (the eigenvalues λ_1 and λ_2 computed by equation (2.42) are two negative real numbers).

unstable equilibrium case

Take parameter $A = 1.8478, B = 8.8257$(which is the case $B > A^2 + 1$). The solution and phase plane plotting is:

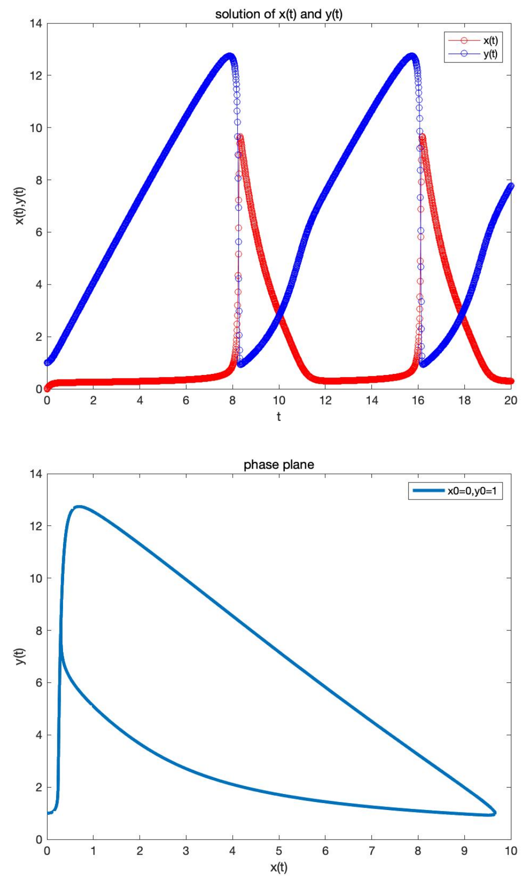


Figure 2.3: unstable equilibrium case

It is clear to see that bifurcation happens in this case(when $B > A^2 + 1$). The equilibrium point is not stable any more and no matter what initial condition we choose, all solutions would diverge from it with respect to the flow of time.

However, what is interesting is that although solutions would diverge from the equilibrium point, they would converge to a new compact invariant object(limit cycle–periodic solution) as $t \to \infty$. To see whether this limit cycle is a global attractor or not, we would take initial condition x_0, y_0 at several points in different parts of the phase plane

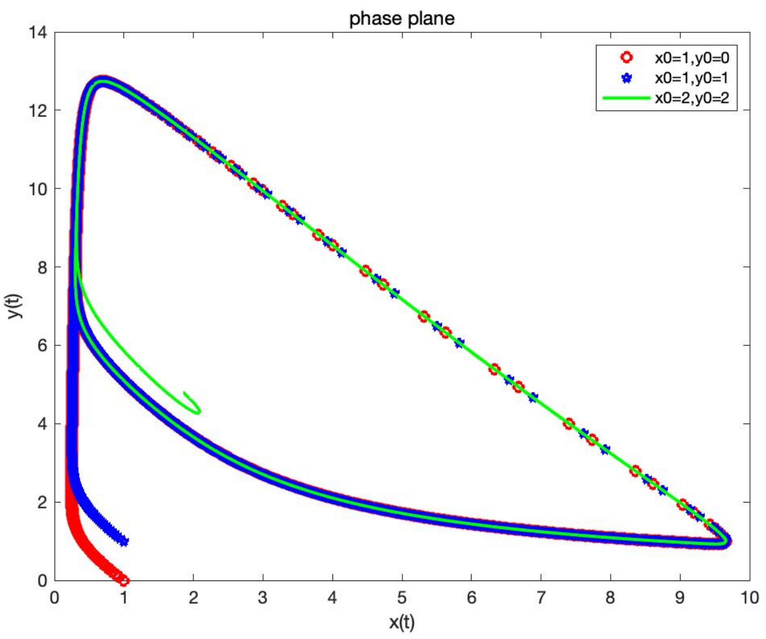


Figure 2.4: global attractor

The numerical result manifest that no matter the trajectories start inside or outside the limit cycle, all solutions still approach the limit cycle with the flow of time. On the basis of such phenomenon, we could expect that this limit cycle might be a global attractor.

However, in most cases, we do not have such a strong theory to get a boundary(For Brusselator case, the boundary is $B > A^2 + 1$) directly for bifurcation. To get a more general method for bifurcation analysis, we would combine the numerical scheme(finite-element,finite-difference) and continuation together, which would be introduced in next section.

Chapter 3

Methodology

This chapter will give a detailed mathematical explanation about continuation methods as well as how it works in bifurcation analysis.

3.1 Solution Branches

Consider the steady-state probelm in the form of:

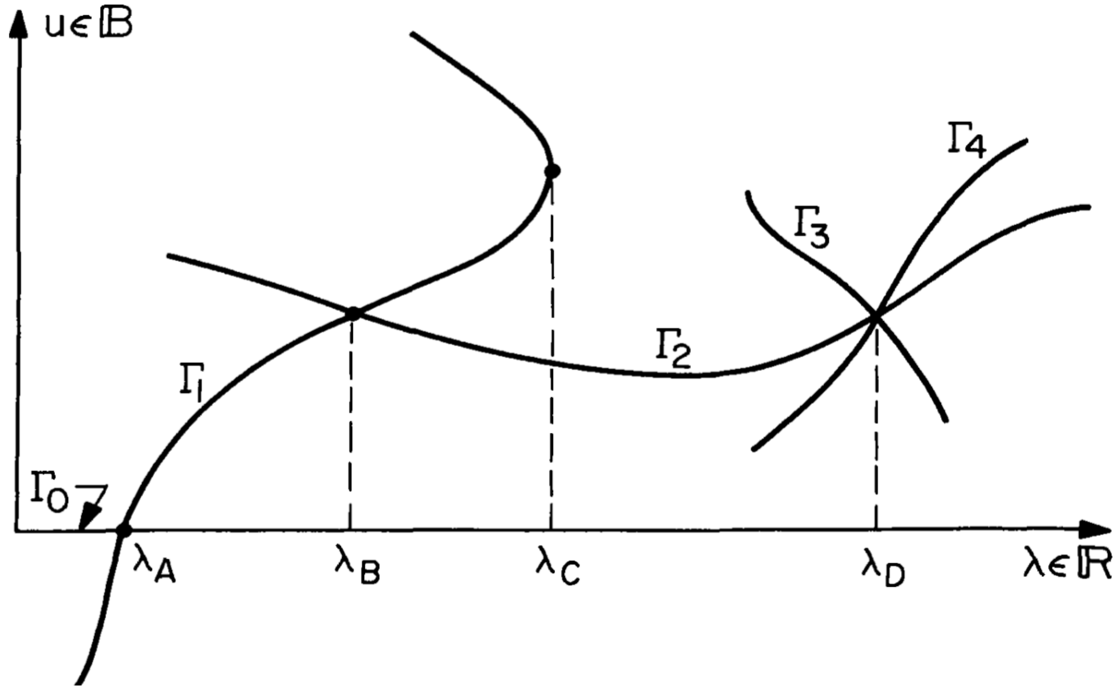
$$\boldsymbol{G}(\boldsymbol{u}, \lambda) = 0 \tag{3.1}$$

where $\boldsymbol{G} : \mathbb{R}^{n+1} \to \mathbb{R}^n$ is the nonlinear equation , $\boldsymbol{u} \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$

Define $\Gamma_{ab} := [\boldsymbol{u}(s), \lambda(s)], s \in [s_a, s_b]$ as the smooth parameterized solution branches such that:

$$\boldsymbol{G}(\boldsymbol{u}(s), \lambda(s)) = 0 \tag{3.2}$$

where s is the parameter of the solution curves, which could be quite arbitrary. In fact, each point on the solution branch is an equilibrium of the dynamical system.



For instance, in this graph, Γ_j, $j = 1, 2, 3, 4$ represents the solution branch curve and the intersection points of different solution branches are where the bifurcation happens.

3.2 Natural Parameter Continuation

This approach would make continuation on the parameter λ directly. In this case, the solution arcs could be defined as $\boldsymbol{u} = \boldsymbol{u}(\lambda)$.
Apply two assumptions [3] here:

- for some $\lambda = \lambda_0$, $u = u_0$ be a solution of system (3.1) such that $G_u(u_0, \lambda_0)$ is nonsingular
- $\exists \rho_0 \in \mathbb{R}$ such that $G(u, \lambda) \in C^1(\Omega)$ where Ω is the sphere region about the point(u_0, λ_0) with radius ρ_0

Then the implicit function theorem could ensure the existence of a unique smooth arc of the solution $u = u(\lambda)$ in Ω [7] through contraction mapping techniques.
In addition, with the smooth assumption, taking the total derivative along the solution branch (3.1) :

$$0 = dG(u, \lambda) = G_u \frac{du}{d\lambda} + G_\lambda \tag{3.3}$$

we get:

$$G_u \frac{du}{d\lambda} = -G_\lambda \tag{3.4}$$

which could suggest a predictor-corrector continuation scheme.

3.2.1 algorithm

predictor

For a given point (u_0, λ_0)(usually comes from a general guess), we use one step Euler's method [1] to construct the predictor:

$$u^{l+1} = u(\lambda_l + \Delta\lambda) = u(\lambda_l) + \Delta\lambda \frac{du}{d\lambda}|_{u=u_l,\lambda=\lambda_l} \tag{3.5}$$

where $l = 0, 1, 2\ldots$ and the upper indices just denote the predictor value and the lower indices denote the final approximation value

corrector

Using Newton's method, we get the corrector in the form of:

$$u^{k+1} = u^k - (G_u^k)^{-1} G^k \quad k = 0, 1, 2\ldots \tag{3.6}$$

As long as this iteration converges, we get our numerical solution of u_1 and repeat such step again and again, the bifurcation diagrams occure

3.2.2 The Criticism of Natural Parameter Continuation

All of these continuation steps might not work as long as the limit point emerges such as $\lambda = \lambda_c$ in the first graph. In this case, making continuation on λ makes no sense. It does need to come up with a new approach to deal with such circumstances, that is, most intuitively, along the arc itself!

3.3 Pseudo Arclength Continuation Method

Instead of thinking the solution branch Γ_{ab} as a function of λ, here we use the parameterized curve to define it such that:

$$\begin{aligned} \Gamma_{ab} :& I \longrightarrow \mathbb{R}^{n+1} \\ & s \longmapsto [\boldsymbol{u}(s), \lambda(s)]^T \end{aligned} \tag{3.7}$$

Hence system (3.1) has been transformed to:

$$\boldsymbol{G}(\boldsymbol{u}(s), \lambda(s)) = 0 \tag{3.8}$$

Although here s could be arbitrary, the best parametrisation should be arc length parametrisation since in this case the Euclidean norm of the tangent vector always equals 1. However, the word "pseudo" in the name of this method means that we do not require the Euclidean norm of the tangent vector to be 1 everywhere on the solution arc, but for another tricky defined norm denoted as ξ-norm.
Then construct the ξ-norm [6] to make Γ_{ab} to be pseudo arclength parametrised solution branch for some $\xi \in (0, 1)$:

$$||\tau||_\xi := \xi||\dot{u}(s)||^2 + (1-\xi)|\dot{\lambda}(s)|^2 = 1 \tag{3.9}$$

where $\tau = [\dot{u}(s), \dot{\lambda}(s)]^T$ is the tangent vector for fixed s
Provided that s_0 is given as well as the point $(u_0, \lambda_0) = (u(s_0), \lambda(s_0))$, and additionally knowing the tangent vector $\tau_0 = (\dot{u}_0, \dot{\lambda}_0) = \dfrac{d}{ds}(u(s), \lambda(s))|_{s=s_0}$ Then the ξ-norm could be transformed to:

$$\begin{aligned} 1 =& \xi||\dot{u}(s)||^2 + (1-\xi)|\dot{\lambda}(s)|^2 \\ 1 =& \xi||\dot{u}_0||^2 + (1-\xi)|\dot{\lambda}_0|^2 \\ s - s_0 =& \xi||\dot{u}_0||||\dot{u}_0||(s-s_0) + (1-\xi)|\dot{\lambda}_0||\dot{\lambda}_0|(s-s_0) \\ s - s_0 =& \xi||\dot{u}_0||(u(s)-u_0) + (1-\xi)|\dot{\lambda}_0|(\lambda(s)-\lambda_0) \end{aligned} \tag{3.10}$$

Hence for some $\xi \in (0, 1)$, the ξ-norm could be redefined as:

$$||\tau||_\xi := \xi||\dot{u}_0||(u(s)-u_0) + (1-\xi)|\dot{\lambda}_0|(\lambda(s)-\lambda_0) - (s-s_0) = 0 \tag{3.11}$$

(3.9) is equivalent to (3.10) as long as the distance of each continuation's step $s - s_0$ is small enough

Hence system (3.1) could be transformed to :

$$H(u(s), \lambda(s)) := \begin{pmatrix} G(u(s), \lambda(s)) \\ N(u(s), \lambda(s), s) \end{pmatrix} = 0 \tag{3.12}$$

where $G(u(s), \lambda(s)) = 0$ ensures the result of solution branches and

$$N(u(s), \lambda(s), s) := \xi||\dot{u}_0||(u(s) - u_0) + (1 - \xi)|\dot{\lambda}_0|(\lambda(s) - \lambda_0) - (s - s_0) = 0$$

guarantees the pseudo arclength parametrisation. In this case, we could **make continuation on the new parameter** s (which is also the pseudo arclength) rather than λ

predictor

Set $(u^1, \lambda^1) = (u_0, \lambda_0) + \tau_0 ds$
here upper indices represent the predictor and lower indices (u_1, λ_1) later would represent the final approximation value

corrector

To use Newton's method, Jacobian matrix of system (3.12) should be firstly calculated:

$$J = \begin{pmatrix} G_u & G_\lambda \\ \xi||\dot{u}_0|| & (1 - \xi)|\dot{\lambda}_0| \end{pmatrix} \tag{3.13}$$

put (u^1, λ^1) into the iteration:

$$\begin{pmatrix} u^{l+1} \\ \lambda^{l+1} \end{pmatrix} = \begin{pmatrix} u^l \\ \lambda^l \end{pmatrix} - A(u^l, \lambda^l)^{-1} H(u^l, \lambda^l) \tag{3.14}$$

where $l = 1, 2, \ldots$
Continuously iterating until convergence, that is the approximation value (u_1, λ_1) of the first continuation step

new tangent vector

Similarly, set $(u^2, \lambda^2) = (u_1, \lambda_1) + \tau_1 ds$ to be the predictor of the second continuation. However, the problem is that what is the new tangent vector τ_1 in the case that (u_1, λ_1) is given.
The most intuitive idea to find the tangent vector of a solution branch is just to

take the total derivative of the origin equation since the meaning of total derivative represents the change of function G and N along the parametrised curve

$$0 = dG(u(s), \lambda(s)) = \partial_u G(u(s), \lambda(s)) \frac{du}{ds} + \partial_\lambda G(u(s), \lambda(s)) \frac{d\lambda}{ds} \tag{3.15}$$

$$0 = dN(u(s), \lambda(s), s) = \partial_u N \frac{du}{ds} + \partial_\lambda N \frac{d\lambda}{ds} + \partial_s N \tag{3.16}$$

At the same time :

$$A(s) \begin{pmatrix} \dot{u}(s) \\ \dot{\lambda}(s) \end{pmatrix} = \begin{pmatrix} \partial_u G(u(s), \lambda(s)) \dot{u}(s) + \partial_\lambda G(u(s), \lambda(s)) \dot{\lambda}(s) \\ \partial_u N \dot{u}(s) + \partial_\lambda N \dot{\lambda}(s) \end{pmatrix} \tag{3.17}$$

By equation (3.15) and (3.16), equation (3.17) could be transformed into:

$$A(s) \begin{pmatrix} \dot{u}(s) \\ \dot{\lambda}(s) \end{pmatrix} = - \begin{pmatrix} 0 \\ \partial_s N \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{3.18}$$

In this case, after convergence of system (3.14), a new point (u_1, λ_1) emerge as well as $A_1 = A(u_1, \lambda_1)$. We could use (3.18) to compute the new tangent vector τ_1:

$$A_1 \tau_1 = \boldsymbol{b} \tag{3.19}$$

where vector $\boldsymbol{b}$ equals $(0, 1)^T$

Repeat such three steps, one solution branch which crosses the point (u_0, λ_0) would emerge

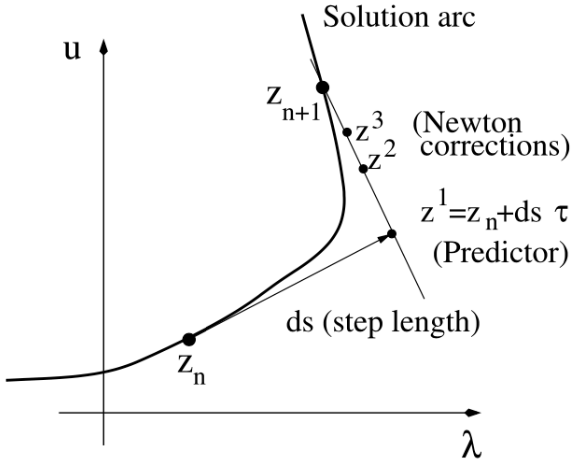


Figure 3.1: one step arclength continuation

3.4 Bifurcation Test

Bifurcation point

A **bifurcation point** is a solution point of equation (3.1) denoted as $(\boldsymbol{u_0}, \lambda_0)$ where the number of solution branches changes when λ crosses λ_0

Bifurcation test function

By first calling arclength continuation, we would get a chain of solutions, such that:

$$(\boldsymbol{u_1}, \lambda_1), (\boldsymbol{u_2}, \lambda_2) \dots (\boldsymbol{u_i}, \lambda_i) \dots$$

A bifurcation point $(\boldsymbol{u_0}, \lambda_0) \in \mathbb{R}^{n+1}$ might occure between $(\boldsymbol{u_{j-1}}, \lambda_{j-1})$ and $(\boldsymbol{u_j}, \lambda_j)$. The Framework is that we need a test function $\tau(\boldsymbol{u}, \lambda)$ evaluated along branch tracing. At the same time, the occure of a bifurcation point is equivalent to test function τ achieving 0:

$$\tau(\boldsymbol{u_0}, \lambda_0) = 0$$

Moreover, this test function τ should be continuous in a sufficiently large interval that includes λ_0. Although it is quite difficult to detect the point at which τ hits zero exactly, it still makes sense to check for a change of sign of the test function such that:

$$\tau(\boldsymbol{u_{j-1}}, \lambda_{j-1})\tau(\boldsymbol{u_j}, \lambda_j) < 0$$

We could visualise such process by a graph:

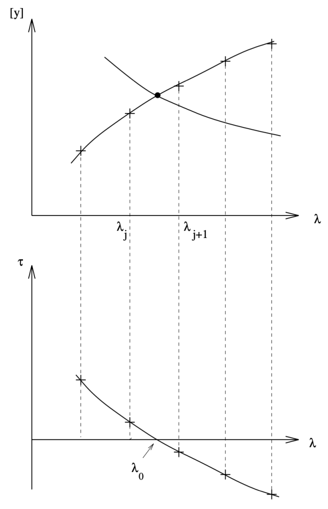


Figure 3.2: Values of a test function τ straddling a zero

The choice of τ

Unfortunately, up to now, there does not exist a perfect choice for test function τ, which means that the missing of some bifurcation points is inevitable.
The most popular choice of τ is the maximum of all real parts of the eigenvalues $\alpha_k + i\beta_k$ of the Jacobian $\dfrac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}(\boldsymbol{u}(\lambda), \lambda)$, such that:

$$\tau(\boldsymbol{u}(\lambda), \lambda) := \max\{\alpha_1, \dots, \alpha_n\}$$

The main advantage for this choice is the physical meaning since the local stability of solution would be ensured when $\tau < 0$. In this case, the change of sign of τ would manifest the stability status of the equilibrium alters. Moreover, if $\boldsymbol{G}(\boldsymbol{u}, \lambda)$ is continuouslly differentiable, then the continuity [11] of test function τ could be guaranteed.
However, such test function cannot signal any bifurcation point at which two unstable half-branches coalesce. The reason is that all real parts of eigenvalues are negative at such bifurcation points and hence τ would never change sign.

Calculating bifurcation point

As long as one bifurcation point has been located in the subinterval

$$\lambda_{j-1} < \lambda_0 < \lambda_j$$

through noticing a change of the sign of τ. The most simple method is just to interpolate the discrete points (λ_k, τ_k) in order to get an approximation of the zero of τ, as shown in figure:

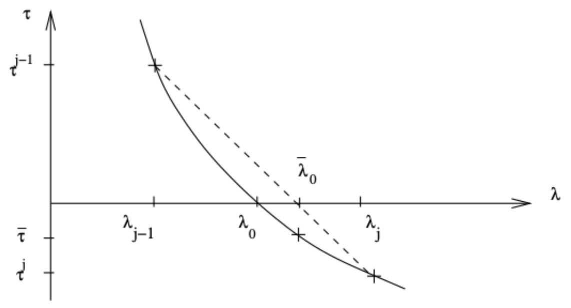


Figure 3.3: Approximating a zero of τ

3.5 Switch Branches

As long as the bifurcation points are calculated, now the problem is how to jump over such points and find the new non-trivial solution branches. First consider equation (3.1) in arclength parameter s:

$$\boldsymbol{G}(\boldsymbol{u}(s), \lambda(s)) = 0 \tag{3.20}$$

where $\boldsymbol{u} \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$
By finite element scheme, we have n unknown nodal values, hence we get n equations

$$\begin{aligned} \boldsymbol{G} :& \mathbb{R}^{n+1} \longrightarrow \mathbb{R}^n \\ & [\boldsymbol{u}(s), \lambda(s)]^T \longmapsto \boldsymbol{G}(\boldsymbol{u}(s), \lambda(s)) \end{aligned} \tag{3.21}$$

such that:

$$\begin{cases} G_1(\boldsymbol{u}(s), \lambda(s)) = & 0 \\ G_2(\boldsymbol{u}(s), \lambda(s)) = & 0 \\ & \vdots \\ G_n(\boldsymbol{u}(s), \lambda(s)) = & 0 \end{cases} \tag{3.22}$$

The overall strategy is finding the new tangent vector of the new solution branch at the bifurcation point and then start continuation.
As mentioned, the most intuitive idea to find the tangent vector of a solution branch is just to take the total derivative of the origin equation since the meaning of total derivative represents the change of function $\boldsymbol{G}$ along the parametrised curve, which could be written in matrix form:

$$\begin{pmatrix} \frac{\partial G_1}{\partial u_1} & \frac{\partial G_1}{\partial u_2} & \cdots & \frac{\partial G_1}{\partial u_n} & \frac{\partial G_1}{\partial \lambda} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \cdots & \ddots & \vdots & \vdots \\ \frac{\partial G_n}{\partial u_1} & \frac{\partial G_n}{\partial u_2} & \cdots & \frac{\partial G_n}{\partial u_n} & \frac{\partial G_n}{\partial \lambda} \end{pmatrix} \begin{bmatrix} \frac{du_1}{ds} \\ \vdots \\ \frac{du_n}{ds} \\ \frac{d\lambda}{ds} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} \tag{3.23}$$

Let $\boldsymbol{X}(s) := \begin{bmatrix} \boldsymbol{u}(s) \\ \lambda(s) \end{bmatrix} \in \mathbb{R}^{n+1}$. Then this Jacobian matrix

$$\begin{pmatrix} \frac{\partial G_1}{\partial u_1} & \frac{\partial G_1}{\partial u_2} & \cdots & \frac{\partial G_1}{\partial u_n} & \frac{\partial G_1}{\partial \lambda} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \cdots & \ddots & \vdots & \vdots \\ \frac{\partial G_n}{\partial u_1} & \frac{\partial G_n}{\partial u_2} & \cdots & \frac{\partial G_n}{\partial u_n} & \frac{\partial G_n}{\partial \lambda} \end{pmatrix}$$

could be denoted as $\dfrac{\partial \boldsymbol{G}}{\partial \boldsymbol{X}}$ with size $n \times n+1$
In addition, denote the tangent vector of the solution branch

$$\begin{bmatrix} \dfrac{du_1}{ds} \\ \vdots \\ \dfrac{du_n}{ds} \\ \dfrac{d\lambda}{ds} \end{bmatrix} \in \mathbb{R}^{n+1}$$

as $\boldsymbol{\tau}$. From this linear system, we get that :

$$Null(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{X}}|_{s=s_0}) \oplus Range(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{X}}|_{s=s_0})^T = \mathbb{R}^{n+1} \tag{3.24}$$

where s_0 is where $\boldsymbol{G}$ achieves its bifurcation points
This is ensured by **Fundamental Theorem Of Linear Algebra** [12]
Generally, for each bifurcation point, there might be m(must strictly greater than 1) non-tangential solution branches intersect, such that [6]:

$$dimN(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{X}}|_{s=s_0}) = m \tag{3.25}$$

In particular, we may firstly focus our attention on the most simple case, that is, only 2 non-tangential solution branches intersect at the bifurcation point and hence $m = 2$
By equation (3.24), we get:

$$\begin{aligned} dimN(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{X}}|_{s=s_0}) &= n+1-dimR(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{X}}|_{s=s_0}) \\ &= n+1-2 = n-1 \end{aligned} \tag{3.26}$$

Hence we get that the **column space** of Jaconbian matrix $\dfrac{\partial \boldsymbol{G}}{\partial \boldsymbol{X}}|_{s=s_0}$ should contain at most $n-1$ linearly independent basis vectors, or equivalently to say, has **rank** $n-1$
Assumption: Suppose that the square matrix $\dfrac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0}$ has the maximal rank, that is

$$\begin{cases} dimR(\dfrac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0}) = n-1 \\ \dfrac{\partial \boldsymbol{G}}{\partial \lambda}|_{s=s_0} \in Range(\dfrac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0}) \end{cases} \tag{3.27}$$

Then we apply **fundamental theorem of linear algebra** again, such that:

$$Null(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0}) \oplus Range(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0})^T = \mathbb{R}^n \tag{3.28}$$

Hence $dimN(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0}) = n - (n-1) = 1$, which is equivalent to say that

$$\begin{aligned} Null(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0}) &= span\{\boldsymbol{\phi_1}\} \\ Null(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0})^T &= span\{\boldsymbol{\psi}\} \end{aligned} \tag{3.29}$$

where $\boldsymbol{\phi_1} \in \mathbb{R}^n$, $\boldsymbol{\psi} \in \mathbb{R}^n$
From assumption $\frac{\partial \boldsymbol{G}}{\partial \lambda}|_{s=s_0} \in Range(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0})$, by the definition of **linear dependence**, we could get : $\exists ! \boldsymbol{\phi_2} \in Null(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0})^{\perp} = Range(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0})^T$, such that:

$$\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0}\boldsymbol{\phi_2} + \frac{\partial \boldsymbol{G}}{\partial \lambda}|_{s=s_0} = 0 \tag{3.30}$$

At the same time, if we take the total derivative of(3.1) and combine it with equation (3.30), we would have:

$$\begin{cases} \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0}\frac{d\boldsymbol{u}}{ds} + \frac{\partial \boldsymbol{G}}{\partial \lambda}|_{s=s_0}\frac{d\lambda}{ds} = 0 \\ \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0}\boldsymbol{\phi_2} + \frac{\partial \boldsymbol{G}}{\partial \lambda}|_{s=s_0} = 0 \end{cases} \tag{3.31}$$

Hence we could write $\frac{\partial \boldsymbol{G}}{\partial \lambda}|_{s=s_0}$ equals $-\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0}\boldsymbol{\phi_2}$ and substitute into the first equation:

$$\begin{aligned} \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0}\frac{d\boldsymbol{u}}{ds} - \frac{d\lambda}{ds}\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0}\boldsymbol{\phi_2} = 0 \\ \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0}(\frac{d\boldsymbol{u}}{ds} - \frac{d\lambda}{ds}\boldsymbol{\phi_2}) = 0 \end{aligned} \tag{3.32}$$

Let $\beta = \frac{d\lambda}{ds} \in \mathbb{R}$, thus the item $\frac{d\boldsymbol{u}}{ds} - \frac{d\lambda}{ds}\boldsymbol{\phi_2}$ belongs to the **Null space** of $\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0}$, such that

$$\frac{d\boldsymbol{u}}{ds} - \beta\boldsymbol{\phi_2} \in Null(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0}) \tag{3.33}$$

Recall equation (3.29), we get:

$$\frac{d\boldsymbol{u}}{ds} - \beta\boldsymbol{\phi_2} = \alpha\boldsymbol{\phi_1} \tag{3.34}$$

for some $\alpha \in \mathbb{R}$.
In this case, we could write the tangent vector $\boldsymbol{\tau}$ in the form of linear combination of two basis vectors:

$$\begin{aligned} \boldsymbol{\tau} = \begin{bmatrix} \frac{d\boldsymbol{u}}{ds} \\ \frac{d\lambda}{ds} \end{bmatrix} = \begin{bmatrix} \beta\boldsymbol{\phi_2} + \alpha\boldsymbol{\phi_1} \\ \beta \end{bmatrix} &= \alpha \begin{bmatrix} \boldsymbol{\phi_1} \\ 0 \end{bmatrix} + \beta \begin{bmatrix} \boldsymbol{\phi_2} \\ 1 \end{bmatrix} \\ &= \alpha\boldsymbol{\Phi_1} + \beta\boldsymbol{\Phi_2} \end{aligned} \tag{3.35}$$

From here, firstly, we could get the result that the dimension of the null space of $\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{X}}|_{s=s_0}$ equals 2, which implies that for any bifurcation point where there are only two non-tangential solution branches intersect, if and only if assumption (3.27) holds. Moreover, two non-tangential solution branches just represent two different pair of coefficients(α and β). Next we just need to find out these two pair coeffients.

coefficients

Differentiating equation (3.23) again, we get:

$$\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{X}}|_{s=s_0}\frac{d^2\boldsymbol{X}}{ds^2}|_{s=s_0}+\frac{\partial^2 \boldsymbol{G}}{\partial \boldsymbol{X}^2}|_{s=s_0}\frac{d\boldsymbol{X}}{ds}|_{s=s_0}\frac{d\boldsymbol{X}}{ds}|_{s=s_0}=0 \tag{3.36}$$

Multiplying this equation by $\boldsymbol{\psi}^T$:

$$\boldsymbol{\psi}^T\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{X}}|_{s=s_0}\frac{d^2\boldsymbol{X}}{ds^2}|_{s=s_0}+\boldsymbol{\psi}^T\frac{\partial^2 \boldsymbol{G}}{\partial \boldsymbol{X}^2}|_{s=s_0}\frac{d\boldsymbol{X}}{ds}|_{s=s_0}\frac{d\boldsymbol{X}}{ds}|_{s=s_0}=0 \tag{3.37}$$

At the same time, the item$\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{X}}|_{s=s_0}\frac{d^2\boldsymbol{X}}{ds^2}|_{s=s_0}$ could be written as:

$$\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{X}}|_{s=s_0}\frac{d^2\boldsymbol{X}}{ds^2}|_{s=s_0}=\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0}\frac{d^2\boldsymbol{u}}{ds^2}|_{s=s_0}+\frac{\partial \boldsymbol{G}}{\partial \lambda}|_{s=s_0}\frac{d^2\lambda}{ds^2}|_{s=s_0} \tag{3.38}$$

Then:

$$\begin{aligned}\boldsymbol{\psi}^T\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{X}}|_{s=s_0}\frac{d^2\boldsymbol{X}}{ds^2}|_{s=s_0} &=\boldsymbol{\psi}^T\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0}\frac{d^2\boldsymbol{u}}{ds^2}|_{s=s_0}+\boldsymbol{\psi}^T\frac{\partial \boldsymbol{G}}{\partial \lambda}|_{s=s_0}\frac{d^2\lambda}{ds^2}|_{s=s_0}\\ &=(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0})^T\boldsymbol{\psi}\frac{d^2\boldsymbol{u}}{ds^2}|_{s=s_0}+(\frac{\partial \boldsymbol{G}}{\partial \lambda}|_{s=s_0})^T\boldsymbol{\psi}\frac{d^2\lambda}{ds^2}|_{s=s_0}\end{aligned} \tag{3.39}$$

By equation (3.29), we have $(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0})^T\boldsymbol{\psi}=0$. In addition, $\frac{\partial \boldsymbol{G}}{\partial \lambda}|_{s=s_0}\in Range(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0})$, $\boldsymbol{\psi}\in Null(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0})^T$, by fundamental theorem of linear algebra, we have:

$$Range(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0})\perp Null(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}|_{s=s_0})^T \tag{3.40}$$

Hence the inner product between vector $\frac{\partial \boldsymbol{G}}{\partial \lambda}|_{s=s_0}$ and $\boldsymbol{\psi}$ equals 0

Combining them together, we get $(\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{X}}|_{s=s_0})^T\boldsymbol{\psi}=0$ and equation (3.37) becomes:

$$\boldsymbol{\psi}^T\frac{\partial^2 \boldsymbol{G}}{\partial \boldsymbol{X}^2}|_{s=s_0}\frac{d\boldsymbol{X}}{ds}|_{s=s_0}\frac{d\boldsymbol{X}}{ds}|_{s=s_0}=0 \tag{3.41}$$

Since $\boldsymbol{X}(s) = \begin{bmatrix} \boldsymbol{u}(s) \\ \lambda(s) \end{bmatrix}$, hence we could rewrite equation (3.41) into:

$$\begin{aligned} \boldsymbol{\psi}^T (\frac{\partial^2 \boldsymbol{G}}{\partial \boldsymbol{u}^2}|_{s=s_0} \frac{d\boldsymbol{u}}{ds}|_{s=s_0} \frac{d\boldsymbol{u}}{ds}|_{s=s_0} + 2\frac{\partial^2 \boldsymbol{G}}{\partial \boldsymbol{u}\partial\lambda}|_{s=s_0} \frac{d\boldsymbol{u}}{ds}|_{s=s_0} \frac{d\lambda}{ds}|_{s=s_0} \\ + \frac{\partial^2 \boldsymbol{G}}{\partial \lambda^2}|_{s=s_0} \frac{d\lambda}{ds}|_{s=s_0} \frac{d\lambda}{ds}|_{s=s_0}) = 0 \end{aligned} \tag{3.42}$$

Substituting $\frac{d\boldsymbol{u}}{ds}|_{s=s_0}$ as well as $\frac{d\lambda}{ds}|_{s=s_0}$ by $\beta\boldsymbol{\phi_2} + \alpha\boldsymbol{\phi_1}$ and β, (3.42) could be transformed into a quadratic equation in the form of:

$$a_{11}\alpha^2 + 2a_{12}\alpha\beta + a_{22}\beta^2 = 0 \tag{3.43}$$

where:

$$\begin{cases} a_{11} = \boldsymbol{\psi}^T \frac{\partial^2 \boldsymbol{G}}{\partial \boldsymbol{u}^2}|_{s=s_0} \boldsymbol{\phi_1}\boldsymbol{\phi_1} \\ a_{12} = \boldsymbol{\psi}^T [\frac{\partial^2 \boldsymbol{G}}{\partial \boldsymbol{u}^2}|_{s=s_0} \boldsymbol{\phi_2} + \frac{\partial^2 \boldsymbol{G}}{\partial \boldsymbol{u}\partial\lambda}|_{s=s_0}] \boldsymbol{\phi_1} \\ a_{22} = \boldsymbol{\psi}^T [\frac{\partial^2 \boldsymbol{G}}{\partial \boldsymbol{u}^2}|_{s=s_0} \boldsymbol{\phi_2}\boldsymbol{\phi_2} + 2\frac{\partial^2 \boldsymbol{G}}{\partial \boldsymbol{u}\partial\lambda}|_{s=s_0} \boldsymbol{\phi_2} + \frac{\partial^2 \boldsymbol{G}}{\partial \lambda^2}|_{s=s_0}] \end{cases} \tag{3.44}$$

However, the second derivative item such like $\frac{\partial^2 \boldsymbol{G}}{\partial \boldsymbol{u}^2}|_{s=s_0}$ in usual is too involved to calculate, hence we need some method to approximate this quantity. Here our choice to calculate this derivative term is to use one-step **Forward Euler** method, such that:

$$\frac{\partial^2 \boldsymbol{G}}{\partial \boldsymbol{u}^2} = \lim_{\epsilon \to 0} \frac{1}{\epsilon} [\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}(\boldsymbol{u}(s) + \epsilon\boldsymbol{\phi_1}, \lambda(s_0)) - \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}(\boldsymbol{u}(s), \lambda(s_0))] \tag{3.45}$$

Since we need an approximation of $\frac{\partial^2 \boldsymbol{G}}{\partial \boldsymbol{u}^2}|_{s=s_0}$ which represents the change of $\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}$ at $s = s_0$ along the tangent direction, hence vector $\boldsymbol{\phi_1}$ would be the most appropriate choice. In this case, equation (3.43) would become

$$a_{11}^{\epsilon}\alpha^2 + 2a_{12}^{\epsilon}\alpha\beta + a_{22}^{\epsilon}\beta^2 = 0 \tag{3.46}$$

Since we have already known one pair of solution, denoted as $\hat{\alpha}$ and $\hat{\beta}$, it would not be difficult to compute the other one pair.

As long as we get the tangent vector of the new branch at the bifurcation point, keep continuation and the new solution branch would emerge.

3.6 Program Implementation–Swift-Hohenberg equation

In this section, the program would be applied for making bifurcation analysis to the Swift-Hohenberg equation (in a form of tutorial). If users want to utilize this program for their own dynamical system problems, we highly recommend them following this example step by step. First of all, a brief introduction to Swift-Hohenberg equation.

3.6.1 Problem Setting

The Swift-Hohenberg equation is a nonlinear fourth order partial differential equation widely used as a model for the research of pattern formation [2]. It is defined as:

$$\frac{\partial u}{\partial t} = \lambda u - (1 + \frac{\partial^2}{\partial x^2})^2 u - u^3 + \nu u^2 \tag{3.47}$$

Then we assume this equation with a homogeneous Neumann boundary conditions such that:

$$\partial_x u|_{x=a} = \partial_x u|_{x=b} = \partial_x \Delta u|_{x=a} = \partial_x \Delta u|_{x=b} = 0 \tag{3.48}$$

To overcome the difficulty brought by the fourth order derivative term, the method we use is to rewrite this 4th order equation as a 2-component second order differential equation system in a consistent way:

$$Let \quad \vec{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{3.49}$$

where

$$\begin{aligned} u_1 &= u \\ u_2 &= \frac{\partial^2 u}{\partial x^2} \end{aligned} \tag{3.50}$$

Hence equation (3.47) could be transfromed into a 2nd order system in the form of $M\partial_t \vec{u} = F(\vec{u}, \Delta\vec{u}, \lambda)$:

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \partial_t \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} -\dfrac{\partial^2 u_2}{\partial x^2} - 2u_2 + (\lambda - 1)u_1 + f(u_1) \\ -\dfrac{\partial^2 u_1}{\partial x^2} + u_2 \end{bmatrix} := F(\vec{u}, \Delta\vec{u}, \lambda) \tag{3.51}$$

where $f(u_1) = -u^3 + \nu u^2$

3.6.2 Construction of G(u,λ)

To implement continuation scheme, we have to find an appropriate form (Recall that function $G(\boldsymbol{u}, \lambda)$(3.1) does not contain any derivative term) to approximate (3.51). Theoretically, both finite-difference and finite-element method could be applied to such a problem. However, due to the time limitation of the project, here only the finite-element method is considered (we might consider to use finite-difference method in future).

Weak formulation

Multiply (3.51) by test function $v \in C^{\infty}[a, b]$ both left hand side and right hand side and integrate:

$$\begin{aligned} \int_a^b \frac{\partial u_1}{\partial t} v dx &= \int_a^b (-\frac{\partial^2 u_2}{\partial x^2} - 2u_2 + (\lambda - 1)u_1 + f(u_1))v dx \\ 0 &= \int_a^b (-\frac{\partial^2 u_1}{\partial x^2} + u_2) v dx \end{aligned} \tag{3.52}$$

For the first equation, by applying integration by part:

$$\int_a^b \frac{\partial u_1}{\partial t} v dx = \int_a^b \frac{\partial u_2}{\partial x}\frac{\partial v}{\partial x} dx - [\frac{\partial u_2}{\partial x} v]_a^b + \int_a^b (-2u_2 + (\lambda - 1)u_1 + f(u_1))v dx \tag{3.53}$$

By homogeneous Neumann boundary conditions (3.48), term $[\frac{\partial u_2}{\partial x} v]_a^b$ will vanish
Same operation for the second equation:

$$\int_a^b \frac{\partial u_1}{\partial x}\frac{\partial v}{\partial x} dx + \int_a^b u_2 v dx = 0 \tag{3.54}$$

Interpolation

Define trial functions:

$$\begin{aligned} u_1 &\simeq \phi_j(x) p_j(t) \\ u_2 &\simeq \phi_j(x) q_j(t) \end{aligned} \tag{3.55}$$

where $j = 1, 2, \dots, n$,
$p_j(t)$ is the nodal value for u_1 at spatial point $x = x_j$,
$q_j(t)$ is the nodal value for u_2 at spatial point $x = x_j$,
ϕ_j is the shape function which determines the geometry between nodal values.
u_1 and u_2 share the same shape function with different nodal values which should be determined by finite-element equation

Applying Galerkin method, set $v_i = \phi_i$. In addition, substitute the trial function (3.55) into weak formulation (3.53)

$$\begin{aligned}\int_a^b \sum_{j=1}^n \phi_i\phi_j dx\dot{p}_j = \int_a^b \sum_{j=1}^n \frac{d\phi_i}{dx}\frac{d\phi_j}{dx}dxq_j + \int_a^b \sum_{j=1}^n \phi_i\phi_j dx(-2q_j + (\lambda-1)p_j) \\ + \int_a^b f(\sum_{j=1}^n \phi_j p_j)\phi_i dx\end{aligned} \tag{3.56}$$

Here we apply the same technique [10] that we used at step (2.9) to approximate the term $\int_a^b f(\sum_{j=1}^n \phi_j p_j)\phi_i dx$:

$$\int_a^b f(\sum_{j=1}^n \phi_j p_j)\phi_i dx \simeq \int_a^b \sum_{j=1}^n \phi_j f(p_j)\phi_i dx = \int_a^b \sum_{j=1}^n \phi_i\phi_j dx f(p_j)$$

Hence equation (3.56) becomes:

$$\int_a^b \sum_{j=1}^n \phi_i\phi_j dx\dot{p}_j = \int_a^b \sum_{j=1}^n \frac{d\phi_i}{dx}\frac{d\phi_j}{dx}dxq_j + \int_a^b \sum_{j=1}^n \phi_i\phi_j dx(-2q_j + (\lambda-1)p_j + f(p_j)) \tag{3.57}$$

Similarly, weak formulation (3.54) could be transformed to:

$$0 = \int_a^b \sum_{j=1}^n \frac{d\phi_i}{dx}\frac{d\phi_j}{dx}dxp_j + \int_a^b \sum_{j=1}^n \phi_i\phi_j dxq_j \tag{3.58}$$

Define $\vec{p} := \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} \in \mathbb{R}^n$, $\vec{q} := \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix} \in \mathbb{R}^n$ and $\tilde{u} := \begin{bmatrix} \vec{p} \\ \vec{q} \end{bmatrix} \in \mathbb{R}^{2n}$

Then we could combine equations (3.57) and (3.58) into a linear system (which is so-called finite-element formulation):

$$\begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix} \begin{bmatrix} \dot{\vec{p}} \\ \dot{\vec{q}} \end{bmatrix} = -\begin{pmatrix} 0 & -K \\ K & M \end{pmatrix} \begin{bmatrix} \vec{p} \\ \vec{q} \end{bmatrix} + \begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix} \begin{bmatrix} -2q_j + (\lambda-1)p_j + f(p_j) \\ 0 \end{bmatrix} \tag{3.59}$$

Define $\mathcal{M} := \begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix}$ and $\mathcal{K} := \begin{pmatrix} 0 & -K \\ K & M \end{pmatrix}$ and set left hand side to be 0 (steady-state solution)

where K and M are the scalar stiffness matrices and mass matrices such that:

$$M_{ij} = \int_a^b \phi_i\phi_j dx, \quad K_{ij} = \int_a^b \frac{d\phi_i}{dx}\frac{d\phi_j}{dx}dx \tag{3.60}$$

In our program, all these matrices are assembled by **oosetfemops.m** function script under the directory **./Swift-Hohenberg-equation**. Here is the code:

```
function swift_hohenberg_equation=oosetfemops(
    swift_hohenberg_equation)
% set stiffness matrix K and mass matrix M
% for Swift–Hohenberg equation as 2nd order system

% scalar stiffness and mass matrix
[K,M,~]=swift_hohenberg_equation.pdeo.fem.assema(
    swift_hohenberg_equation.pdeo.grid,1,1,1);

% 2nd order system stiffness
swift_hohenberg_equation.mat.K=[[0*K -K];[K M]];
% 2nd order system mass matrix (here is singular)
swift_hohenberg_equation.mat.M=[[M 0*M];[0*M 0*M]];
```

Remark: Since the shape functions ϕ_i could be widely used in most problems, hence this program preassemble the scalar mass M and stiffness matrices K under the folder **finite_element_geometry**. In this case, users do not need to redefine these two matrices each time when they start a new problem. In addition, this program is object oriented programming. Here variable **swift_hohenberg_equation** is a pde (here is sh equation) object (structure) carrying all necessary information.

Then we construct the nonlinear term $\begin{bmatrix} -2q_j + (\lambda - 1)p_j + f(p_j) \\ 0 \end{bmatrix}$ by **nonlinearf.m** function script:

```
function f=nonlinearf(swift_hohenberg_equation,u)
% SH "nonlinearity" for the 2nd-order system formulation
%
% INPUTS
% u: solution of 2nd order pde system such that u=[u1;u2;par]
%
% OUTPUTs
% f: nonlinear part of the 2nd order pde system such that f=[f1;f2]

% split parameters from u
par=u(swift_hohenberg_equation.nu+1:end);
lam=par(1);
nu=par(2);

% split u1 and u2
n=swift_hohenberg_equation.nu/2;
u1=u(1:n);
u2=u(n+1:2*n);

```

```
% construct the nonlinear part in equation
f1=(lam-1)*u1+nu*u1.^2-u1.^3-2*u2;
f2=0*u2;
f=[f1;f2];
```

From now on, with the construction of system mass $\mathcal{M}$ and stiffness $\mathcal{K}$ matrices as well as the nonlinear terms, function (3.1) $\boldsymbol{G}(\boldsymbol{u}, \lambda)$ as well as its Jocobian could be obtained by **G.m** and **Gu.m** function scripts:

```
function r=G(swift_hohenberg_equation,u)

f=nonlinearf(swift_hohenberg_equation,u);
r=swift_hohenberg_equation.mat.K*u(1:
    swift_hohenberg_equation.nu)-swift_hohenberg_equation.mat
    .M*f;
```

Initialization

What should be done in the next stage is to initialise the solution (user should set the domain of differential equation, define the discrete mesh points for interpolating as well as give an appropriate initial guess to start continuation).

For Swift-Hohenberg equation, zero branch is chosen to be the initial branch of the starting point of continuation. Such a work is done by **initialise_pde.m** function script, it is strongly recommended for users to open this file and follow the structure when they restart a new problem.

Chapter 4

Results

Based on the above preparation (see the last section of Chapter 3), now users could create a command line interface and start continuation schemes. This chapter will present some bifurcation as well as solution diagrams of the Swift-Hohenberg equation obtained by the program as well as a validation check.

4.1 Example Outputs 1

First of all, set domain and initialising the solution and parameters:

```
keep pphome;
close all;
swift_hohenberg_equation=[];

% initialise and set up zero-branch (trivial solution) on long domain
% set domain [-20pi,20pi]
lx=20*pi;
nx=round(30*lx);
% set parameters
lam=-0.05;
nu=2;
par=[lam;nu];
swift_hohenberg_equation=initialise_pde(
    swift_hohenberg_equation,nx,lx,par);
```

Here we fixed parameter ν to be 2 in equation (3.47) since our program could only make continuation on one parameter at one time. Set domian of the problem to be $[-20\pi, 20\pi]$, initial continuation parameter λ to be -0.05.

As long as the successful construction of the initial branch, we start to find some bifurcation points on this trivial branch by calling **find_bifurcation_point.m** function (all functions relevent to continuation are under the folder **./continuation_functions/numerical_continuation_function**)

```
swift_hohenberg_equation=find_bifurcation_point(
    swift_hohenberg_equation,4);
```

Here 4 represents the number of bifurcation points we expect to find on the trivial branch. The result is:

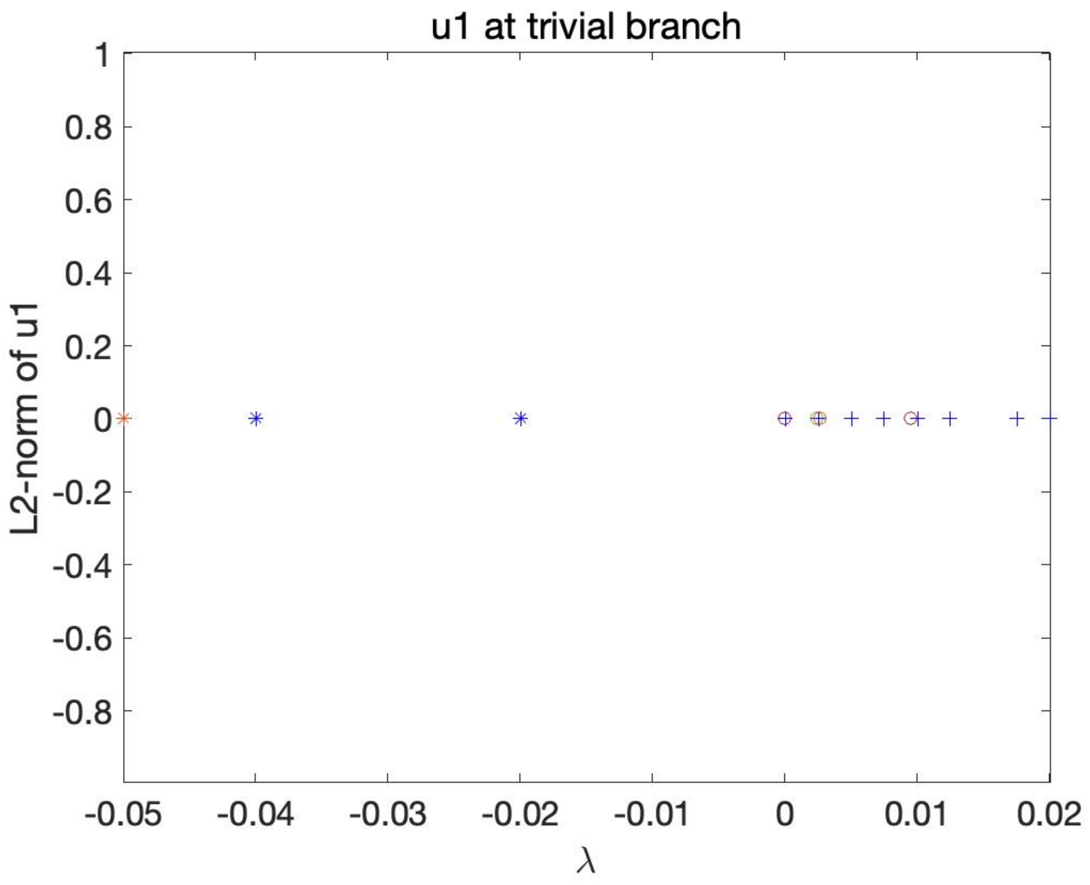


Figure 4.1: bifurcation detection

It is clear to see that four bifurcation points (labeled by small circles) are detected successfully. Save all these points and turn to the first one. By using the switching branch technique (in our program, function script **switch_branch.m**) we have introduced in Chapter 3, the tangent vector of new branch at the first bifurcation point could be got. In this case, starting continuation at the bifurcation point by the new tangent vector and the non-trivial solution branch could be obtained. Here is the code:

```
% Turing-branches
swift_hohenberg_equation=switch_branch('Example1/trivial','
   bpt1','Example1/branch1',0.01);
swift_hohenberg_equation=numerical_continuation(
   swift_hohenberg_equation,40);
```

Here ’bpt1’ means the first bifurcation point we get in last step by calling **find_bifurcation_point.m** function. We switch from the ’trivial branch’ at ’bpt1’ to the first non-trivial branch (’branch1’) and then make continution with 40 steps as well as step size 0.01. The result is:

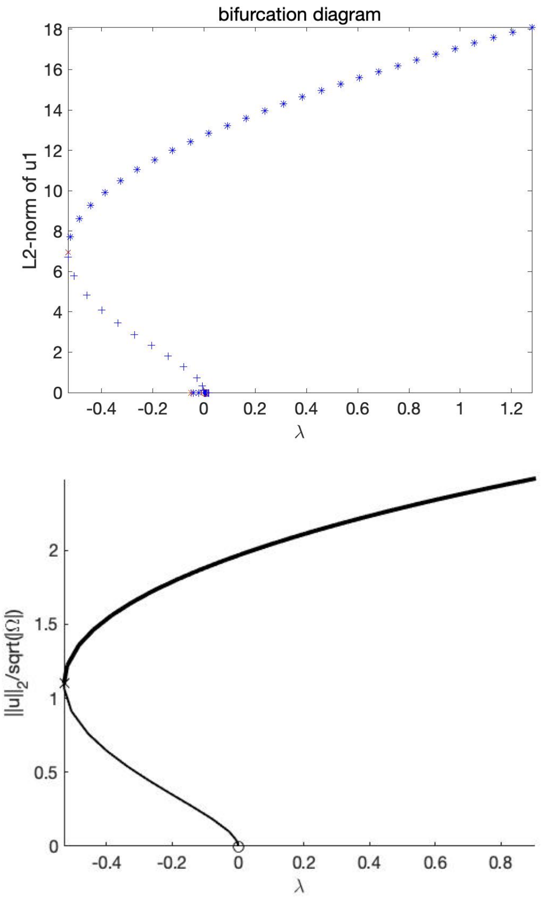


Figure 4.2: bifurcation diagram

A non-trivial branch is obtained successfully.

4.2 Example Outputs 2 (Bifurcation on Bifurcation)

Following the above procedure, we create a new command line interface (**cmds2.m**) to study the new solution branches bifurcated from the old non-trivial bifurcation branches.

To reduce the complexity of computation, we shrink the domain in half (from $[-20\pi, 20\pi]$ to $[-10\pi, 10\pi]$). The procedure of initialization and finding the first non-trivial branch is quite similar to Example-1 and user could read the code in **cmds2.m** file.

Then we need to find some bifurcation points on the old non-trivial branch (named as "branch1" in **cmds2.m**):

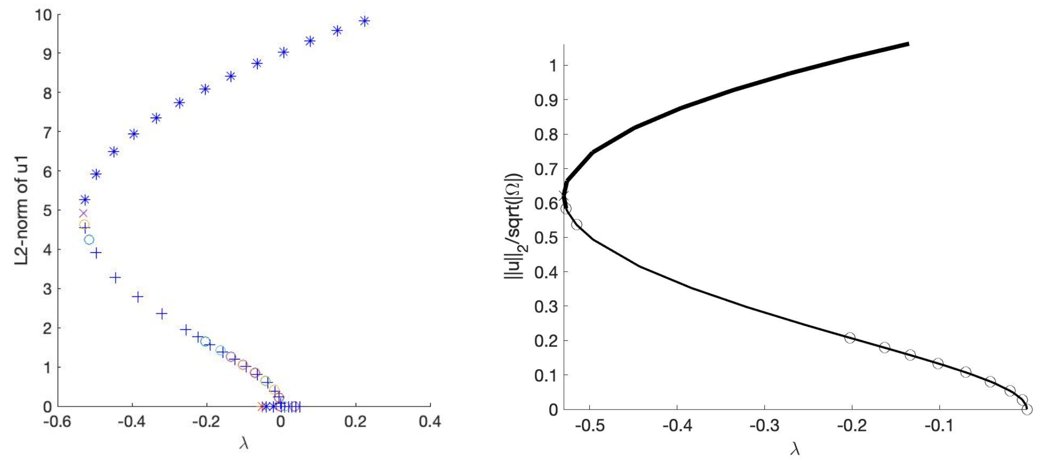


Figure 4.3: bifurcation detection on old branch

From the right graph, it is clear to see that some bifurcation points on this non-trivial branch are detected successfully (labeled by small black circles). We record the first and second bifurcation points.

What should be done in the next step is just to repeat the same operation as shown in Example-1 (turn to the first and second bifurcation point **respectively**, switching branch and making continuation). The result is:

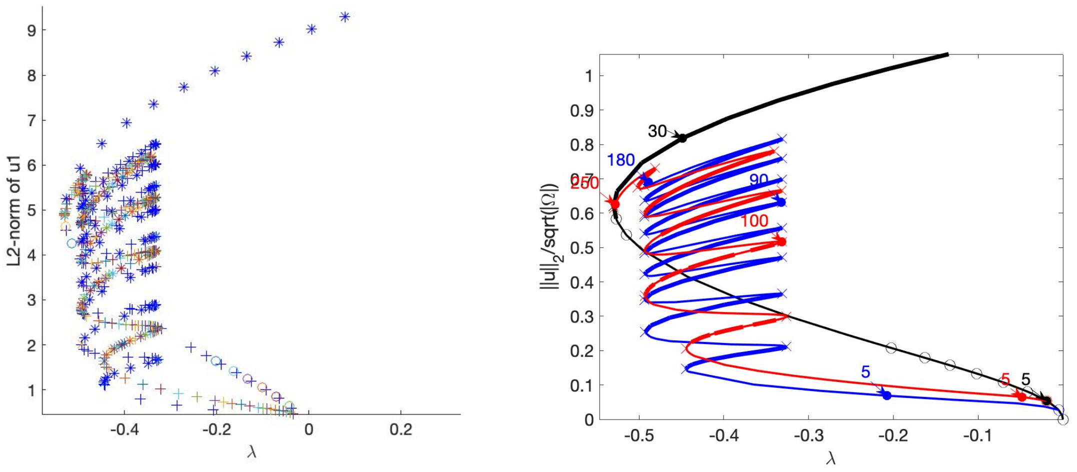


Figure 4.4: bifurcation diagram of 3 non-trivial branches

Initially, both new branches (the blue and red curves) diverge from the old solution branch (black curve) and then rise in the shape of a spiral with radius around 0.05. However, as the proceeding of continuation, these two new solution branches have a tendency to return to the original branch. To verify it, we will plot all labeled points on this bifurcation diagram in the next section called solution plotting.

4.3 Solution Plotting

Firstly, we plot solutions on the black non-trivial branch (called ”branch1” in **cmds2.m**).

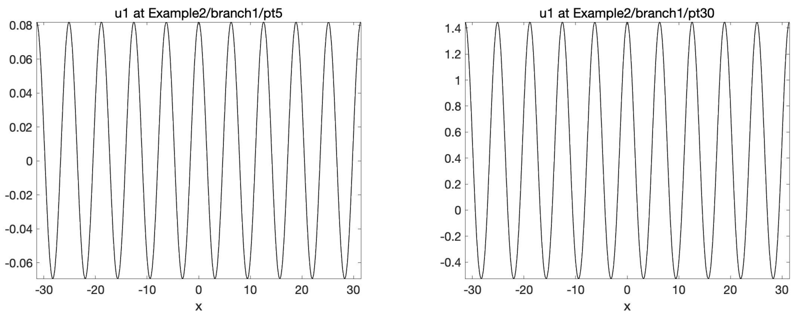


Figure 4.5: solution plotting on branch1

Remark
Both these two points ”pt5”, ”pt30” are labeled on the black curve (branch1) of the bifurcation diagram 4.4.

From this figure 4.5, we could expect that solutions are spatially periodic and time invariant (since what we compute is steady-state solutions) on ”branch1”.

Then we take a look at the solutions on the blue curve (named as subranch1 in **cmds2.m**) of 4.4:

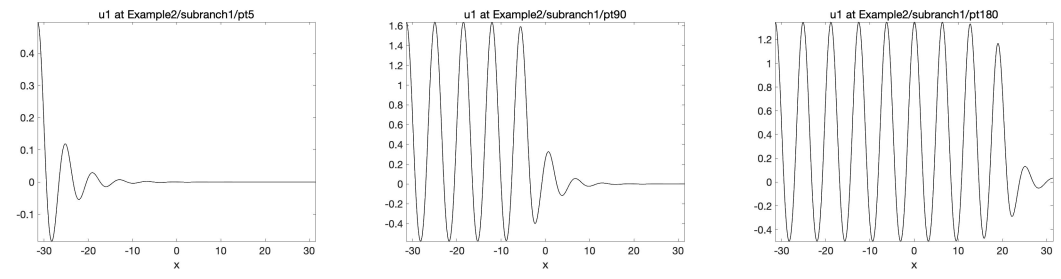


Figure 4.6: solution plotting on subranch1

For solutions at the initial stage of continuation, there are only a few humps that occurred at the left end side of the domain with huge different amplitudes, which is a reasonable change brought by the bifurcation. However, with the proceeding continuation, more and more humps with the same peak appear along the positive direction of the x-axis. We could expect solutions on the blue curve (subranch1) to be spatially periodic again at the end part of the branch, which is consistent with what we observe in the bifurcation diagram 4.4 of the previous section.

Finally, let us see solutions on the red curve (named as subranch2 in **cmds2.m**) of 4.4:

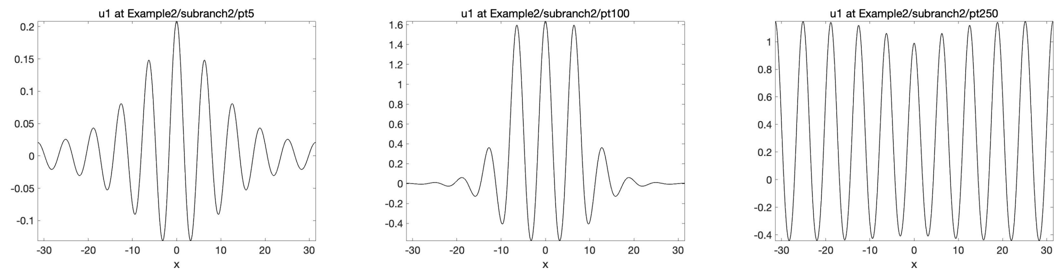


Figure 4.7: solution plotting on subranch2

Overall, solutions on this branch are symmetric about the origin point while humps located close to the middle of the domain usually have a higher peak (except the tail of the branch). Moreover, with more and more humps with the same peak appearing from the middle of the domain as the proceeding of continuation, solutions are getting closer to spatially periodic.

Summary

From the above numerical solution plotting and analysis, users could obtain a systematic and regular description as well as an intuitive feeling of the solutions of the Swift-Hohenberg equation located on different bifurcation branches, which is the main contribution of bifurcation analysis and our program to the dynamical system. Following the above procedures, users could utilize this program to study their own dynamical system problems.

4.4 Validation Check

The most direct way to check the correctness of the numerical solutions obtained by our program is just to compare them with the exact solution. In this section, we utilize the constant branch of the Allen-Cahn equation (see example (2.1)) to check the validation of our program (since it is easy to compute). Set domain to be $[-5, 5]$ and fixed parameters c and γ in the equation to be 1. Here is the constant branch obtained by our program:

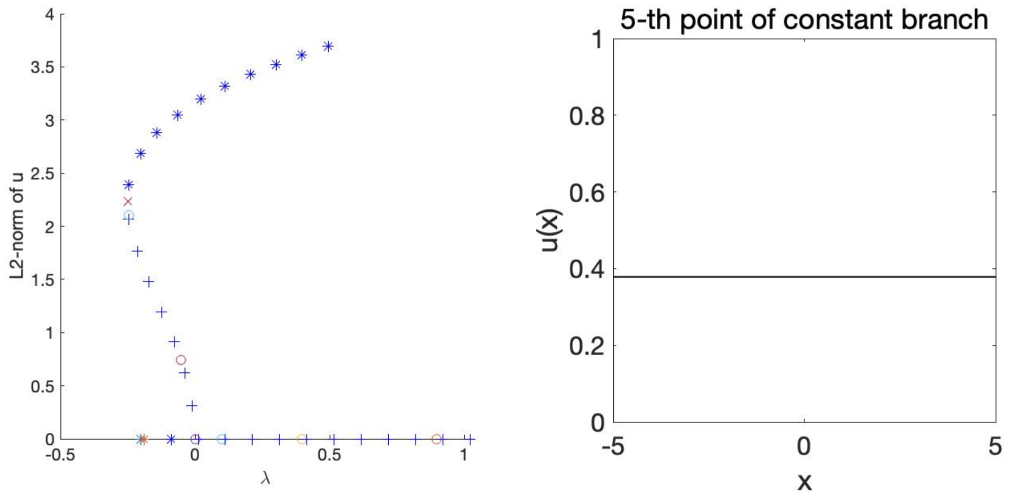


Figure 4.8: constant branch

On the other hand, the exact solution of the constant branch should be in the form of:

$$\begin{aligned} u &= \pm\sqrt{\frac{1 \pm \sqrt{1+4\lambda}}{2}} \\ ||u||_2 &= (\int_{-5}^{5} |u|^2 dx)^{0.5} = \sqrt{10}|u| = \sqrt{5(1 \pm \sqrt{1+4\lambda})} \end{aligned} \tag{4.1}$$

which could be plotted by Matlab:

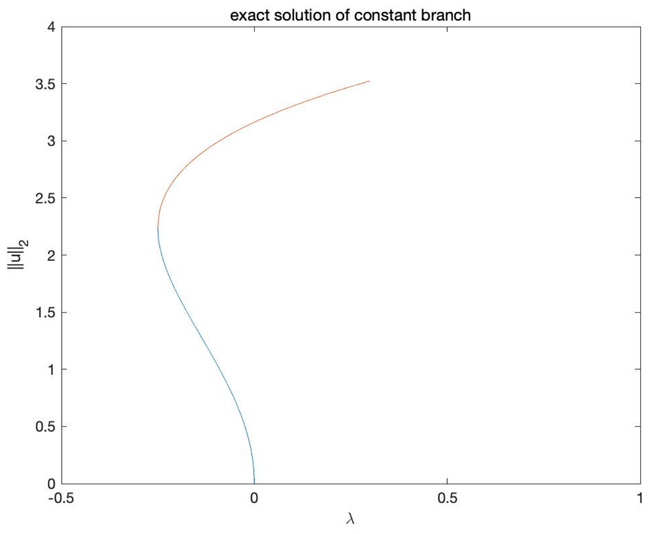


Figure 4.9: exact solution of constant branch

From the above graphs, figure 4.8 (numerical solution) is almost consistent with figure 4.9 (exact solution), proving that our program makes perfect sense.

Coding implementation: see directory **./main_programme/testing_sample/Allen-Cahn-equation**

Chapter 5

Conclusion

This Chapter summarizes the main contribution and results of this project. Then make conclusions for implications, some potential limitations as well as recommendations.

5.1 Summary and Study Implication

To sum up, this project combines numerical time integration approaches (finite-element schemes and finite-difference methods) and numerical continuation path-following schemes together to study the bifurcation analysis of some dynamical systems which may depend on parameters with physical meanings.

On the one hand, numerical time integration approaches can give a good approximation of any derivative terms and then transform differential equations into nonlinear equations which only depend on time, spatial variables, and parameters.

On the other hand, numerical continuation path-following schemes offer an efficient way to solve any nonlinear equation depending on the parameters.

Through the cooperation of these two methods, this dissertation provides readers with a systematic and feasible approach (in practice, a Matlab program) to describe the trajectory flow, solution, and bifurcation diagrams of a dynamical system. Moreover, the more important significance of this dissertation is to help any reader understand the mathematical principle behind this program. Not only could readers be able to run the program, but also they are supposed to know the basic math logic behind it.

5.2 Limitations

This project leads to the following limitations:

First of all, due to the construction of the finite-element geometry in this project, the program could only approximate the derivative terms of the 1D dimension, which means that our work in this project could only be applicable to the dynamical system problems whose spatial variable belongs to the real line. However, most problems in the real world might be 2-dimension or 3-dimension. For instance, even though the Swift-Hohenberg equation mentioned in this dissertation, it also has its 2D and 3D forms.

In addition, as we see at the end of Chapter 3, we have to fix the value of parameter ν since our program could only make continuation on one parameter at one time. However, most problems in science and technology contain at least 2-3 parameters. Making continuation on one parameter and fixing all the others is not efficient.

5.3 Recommendation

Based on the limitations, the recommendations could be similarly summarized into two aspects:

To solve the first limitation, it is suggested to separate the domain of higher dimension into small regular pieces of elements (for instance, triangle or rectangle in higher dimensional spaces) just like what we have done in one dimension. Then, instead of using integration by part, by applying Green's second identity [5] to construct the weak formulation and constructing the mass and stiffness matrices step by step. In future development, it is also necessary to add finite-difference methods to the program for approximating higher dimension derivative terms.

As for the problem of multi-parameters, a future plan is to optimize the continuation algorithm (make continuation on a parameter vector rather than on just one real parameter), making it possible for the program to realize multi-parameter continuation.

Reference List

[1] Eugene L Allgower and Kurt Georg. *Introduction to numerical continuation methods*. SIAM, 2003.

[2] Parisa Bakhtiari, Saeid Abbasbandy, and Robert A Van Gorder. Reproducing kernel method for the numerical solution of the 1d swift–hohenberg equation. *Applied Mathematics and Computation*, 339:132–143, 2018.

[3] Louis Bauer, Edward L Reiss, and Herbert B Keller. Axisymmetric buckling of hollow spheres and hemispheres. Technical report, NEW YORK UNIV NY COURANT INST OF MATHEMATICAL SCIENCES, 1969.

[4] Xiaobing Feng and Andreas Prohl. Numerical analysis of the allen-cahn equation and approximation for mean curvature flows. *Numerische Mathematik*, 94(1):33–65, 2003.

[5] M Fernández-Guasti. Green's second identity for vector fields. *International Scholarly Research Notices*, 2012, 2012.

[6] Herbert B Keller and KELLER HB. Numerical solution of bifurcation and nonlinear eigenvalue problems. 1977.

[7] Steven George Krantz and Harold R Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002.

[8] Bernd Krauskopf, Hinke M Osinga, and Jorge Galán-Vioque. *Numerical continuation methods for dynamical systems*, volume 2. Springer, 2007.

[9] Bing Liu, Yujuan Zhang, and Lansun Chen. The dynamical behaviors of a lotka–volterra predator–prey model concerning integrated pest management. *Nonlinear Analysis: Real World Applications*, 6(2):227–243, 2005.

[10] Jens DM Rademacher and Hannes Uecker. The oopde setting of pde2path–a tutorial via some allen-cahn models, 2018.

[11] Rüdiger Seydel. *Practical bifurcation and stability analysis*, volume 5. Springer Science & Business Media, 2009.

[12] Gilbert Strang. The fundamental theorem of linear algebra. *The American Mathematical Monthly*, 100(9):848–855, 1993.

[13] Endre Süli and David F Mayers. *An introduction to numerical analysis*. Cambridge university press, 2003.

[14] Barna Szabó and Ivo Babuška. *Finite element analysis*. John Wiley & Sons, 1991.

[15] P Yu and AB Gumel. Bifurcation and stability analyses for a coupled brusselator model. *Journal of Sound and vibration*, 244(5):795–820, 2001.

[16] Tomasz G Zieli. Introduction to the finite element method. 1992.

[17] Olgierd Cecil Zienkiewicz, Robert Leroy Taylor, Perumal Nithiarasu, and JZ Zhu. *The finite element method*, volume 3. McGraw-hill London, 1977.