university of groningen / faculty of science and engineering / mathematics and applied mathematics

Comparing two approaches for singular systems occurring during bifurcation analysis

Bachelor's Project Applied Mathematics

July 2021

Student: W.Guo

First supervisor: dr.ir. F.W. Wubs

Second assessor: dr.ir. B. Besselink

Abstract

During bifurcation analysis, singular systems occur at various places in the algorithm. In this thesis, we consider the Jacobian matrix that becomes singular at a bifurcation point and to the shifted matrix occurring in the JDQZ for the eigenvalue problem, which becomes singular on convergence. The former singularity will be studied for the one-dimensional Bratu problem and the latter for the Rayleigh-Bénard problem. In both cases, we apply two different approaches to solve the singular system and explore the influence of various parameters on the results.

Keywords: Bifurcation analysis; Bratu problem; Rayleigh-Bénard problem; Pseudo-arclength continuation; JDQZ;

Contents

1 Introduction

In this thesis, techniques for the bifurcation and linear stability analysis will be introduced. Also, these techniques will be applied to one-dimensional Bratu problem and Rayleigh-Bénard problem. Bifurcation analysis is the mathematical study of changes of a given problem, such as the solutions of differential equations. A bifurcation occurs when a small smooth change made to the parameter values (the bifurcation parameters) of a system causes a sudden qualitative change in its behaviour.

This chapter is organised as follow. First, we shall introduce two problems we want to solve, one-dimensional Bratu problem and Rayleigh-Bénard problem. Then we shall briefly introduce the methods we use and the singular systems encountered. Finally, an overview of the thesis will be presented.

1.1 One-dimensional Bratu problem

The one-dimensional Bratu problem is an elliptic nonlinear partial differential equation problem of the form (1.1) [1]:

$$\begin{aligned} &u_{xx} + Ce^u = 0, \quad 0 \le x \le 1 \\ &u(0) = u(1) = 0 \end{aligned} \tag{1.1}$$

where $C > 0$ is a parameter of the system.

We can simply rewrite it as,

$$\Phi(u, C) = 0$$

This differential equation appears in a variety of applications, such as fuel ignition models for thermal combustion, radiation heat transfer, thermal reactions, Chandrasekhar model for universe expansion, chemical reactor theory, and nanotechnology. At the same time, it is also widely used as a test problem for many numerical algorithms.

1.2 Rayleigh-Bénard problem

The Rayleigh-Benard problem is a differential equation of the form

$$\begin{aligned} Pr^{-1}(\frac{\partial u}{\partial t} + u \cdot \nabla u) &= -\nabla p + \nabla^2 u + Ra \cdot T \cdot e_3 \\ \nabla \cdot u &= 0 \\ \frac{\partial T}{\partial t} + u \cdot \nabla T &= \nabla^2 T \end{aligned} \tag{1.2}$$

with boundary conditions

$$\begin{aligned} z=0 \quad &: \quad T=1; \frac{\partial u}{\partial z}=\frac{\partial v}{\partial z}=w=0 \\ z=1 \quad &: \quad \frac{\partial u}{\partial z}=Ma\frac{\partial T}{\partial x}; \frac{\partial v}{\partial z}=Ma\frac{\partial T}{\partial y}; w=0; \frac{\partial T}{\partial z}=BiT \\ x=0, A_x \quad &: \quad u=v=w=\frac{\partial T}{\partial x}=0 \\ x=0, A_y \quad &: \quad u=v=w=\frac{\partial T}{\partial y}=0 \end{aligned} \tag{1.3}$$

where Pr, Ra, Ma, Bi are four dimensionless parameters. In addition, two aspect ratios $A_x = L/D$ and $A_y = B/D$.

The Rayleigh-Bénard convection problem has been widely investigated because of its fundamental interest in relation to the evolution of flow patterns and the onset of unsteadiness. Many engineering problems related to thermal transport in crystal growth, solar collectors, buildings and nuclear reactor core insulation depend on this type of natural convection[2].

1.3 Singular systems emerging from bifurcation analysis

In general, for bifurcation analysis, one can use two methods: the time integration approach and the continuation approach. In this thesis, the continuation approach is used to obtain the solution of the problems. In addition, Jacobian Davidson QZ method is used to assess the linear stability of the solution.

Before solving our problems, we need to discretize them using finite difference method. Then, the resulting problem can be written in the general form

$$\Phi(u, C) = \mathcal{L}u + \mathcal{N}(u) = 0 \tag{1.4}$$

where u is the solution of original problems, M the mass matrix, $\mathcal{L}$ the discretized linear operator, $\mathcal{N}$ the discretized nonlinear operator.

Natural parameter continuation method is a simple method for bifurcation analysis. One of advantages is that it could be viewed as a black box, since only an initial solution is required. However, if a turning point is encountered with the increase of parameter value during natural parameter continuation, the method will fail. Hence, for problems with turning points, pseudo-arclength continuation method needs to be used.

In each continuation step, one can solve two different kinds of linear system. One is of the form

$$\begin{bmatrix} \Phi_u(u^k, C^k) & \Phi_C(u^k, C^k) \\ r_u(u^k) & r_C(C^k) \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta C \end{bmatrix} = \begin{bmatrix} -\Phi(u^k, C^k) \\ -r(u^k, C^k) \end{bmatrix} \tag{1.5}$$

This system is non-singular at the bifurcation point, however, it is much more computationally intensive.

The other is of the form

$$\begin{cases} \Phi_x y &= \Phi_\mu \\ \Phi_x z &= -\Phi \end{cases} \tag{1.6}$$

Compared with Equation 4.9, Equation 1.6 is more simple. However, it is singular at the bifurcation point, which means that the algorithm may fail when it is close to the bifurcation point. In Chapter 3, we compare the two methods and examine the effect of each parameter in the method on the results.

Once a solution $\overline{u}$ is computed, its linear stability needs to be assessed. This can be done by solving a generalised eigenvalue problem of the form

$$\Phi_u(\overline{u})\hat{u} = \lambda M\hat{u} \tag{1.7}$$

where $u(t) = \hat{u}e^{\lambda t} + \overline{u}$.

This generalised eigenvalue problem can be rewritten in the form

$$(\beta A - \alpha B)u = 0 \tag{1.8}$$

When all eigenvalues of the generalised eigenvalue problem have negative real part, then the solution is linearly stable. Otherwise, the solution is unstable.

In this thesis, Jacobi Davidson QZ(JDQZ) method[3] is used to solve the generalised eigenvalue problem. It turns out that it is an efficient method.

In each JDQZ step, one can solve two different kinds of correction equations. One is of the form

$$z \perp u, \quad (I - qq^*)(A - vB)(I - uu^*)z = 0 \tag{1.9}$$

where v is the approximate eigenvalue, $q = Bu/\|Bu\|$.

Notice that, this system is singular at the bifurcation point, which means that the algorithm may fail when we are close to the real bifurcation point.

The other is of the form

$$\begin{bmatrix} A - vB & q \\ u^* & 0 \end{bmatrix} \begin{bmatrix} z \\ \alpha \end{bmatrix} = - \begin{bmatrix} r \\ 0 \end{bmatrix} \tag{1.10}$$

Compared with the previous correction equation, one advantage of this correction equation is that it is non-singular at the bifurcation point. In Chapter 4, we also compare the two methods and examine the effect of each parameter in the method on the results.

1.4 Overview of this thesis

In this chapter, we first introduce the governing equations of one-dimensional Bratu problem and Rayleigh-Bénard problem, then briefly discuss the methods used to solve these problems and introduce the singular systems occurring during bifurcation analysis. In Chapter 2, we introduce the used algorithms in detail. First, finite difference method that is used to discretized differential equations is presented. Next, the description of pseudo-arclength continuation is given. Also, the preconditioner we used to speed up the algorithms' convergence is introduced. In Chapter 3, we present numerical results of one-dimensional Bratu problem, not only the effect of several parameters but also the performance of our program. In Chapter 4, we use Jacobi Davidson QZ(JDQZ) method to check the linear stability of the solution. The last chapter outlines conclusions.

2 Methodology

In this chapter, we introduce the methods we use, like finite difference method, pseudo-arclength continuation and preconditioner.

2.1 Finite difference method

In this section, we introduce the finite difference method used to discretize differential equations. The finite difference method is a widely used and very effective discretization method. The core idea of the method comes from Taylor's formula[4].

Suppose we want to discretize a partial differential equation $\mathcal{L}(u) = 0$ defined on the interval $[0, 1]$.

First, we divide the interval $[0, 1]$ into N equal parts, and each point on the interval is defined as:

$$x_i = ih, i = 0, 1, 2, \cdots, N, \quad h = \frac{1}{N}$$

Further, we can define the value of the function to be solved u at each point on the interval:

$$u_i = u(x_i), i = 0, 1, 2, \cdots, N$$

Then, using Taylor's formula, one can get the finite difference formula of the derivative of u at the point x_i,

$$u_{i+1} = u(x_i + h) = u_i + \left.\frac{\mathrm{d}u}{\mathrm{d}x}\right|_{x_i} + \left.\frac{h^2}{2!}\frac{\mathrm{d}^2u}{\mathrm{d}x^2}\right|_{x_i} + \left.\frac{h^3}{3!}\frac{\mathrm{d}^3u}{\mathrm{d}x^3}\right|_{x_i} + \left.\frac{h^4}{4!}\frac{\mathrm{d}^4u}{\mathrm{d}x^4}\right|_{x_i+\zeta_1 h} \tag{2.1}$$

as well as

$$u_{i-1} = u(x_i - h) = u_i - h\left.\frac{\mathrm{d}u}{\mathrm{d}x}\right|_{x_i} + \left.\frac{h^2}{2!}\frac{\mathrm{d}^2u}{\mathrm{d}x^2}\right|_{x_i} - \left.\frac{h^3}{3!}\frac{\mathrm{d}^3u}{\mathrm{d}x^3}\right|_{x_i} + \left.\frac{h^4}{4!}\frac{\mathrm{d}^4u}{\mathrm{d}x^4}\right|_{x_i-\zeta_2 h} \tag{2.2}$$

Adding and rearranging the formula (2.1) and the formula (2.2), one can get

$$\left.\frac{\mathrm{d}^2u}{\mathrm{d}x^2}\right|_{x_i} = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} - \left.\frac{h^2}{12}\frac{\mathrm{d}^4u}{\mathrm{d}x^4}\right|_{x_i+\zeta h}, \quad \zeta \in [-1, 1] \tag{2.3}$$

If we ignore the $\mathcal{O}(h^2)$ term, then we call the right side of the formula (2.3) the second-order central difference formula of the second-order derivative function , which is

$$\left.\frac{\mathrm{d}^2u}{\mathrm{d}x^2}\right|_{x_i} = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} \tag{2.4}$$

Similarly, we can also get the second-order central difference formula of the first-order derivative function, namely

$$\left.\frac{\mathrm{d}u}{\mathrm{d}x}\right|_{x_i} = \frac{u_{i+1} - u_{i-1}}{2h}$$

Through the first-order derivative function, the second-order difference formula of the second-order derivative function, and other difference formulas, we can convert the partial differential equation into an algebraic equation, and use the method of solving algebraic equations to solve it, thereby obtaining an approximation of the partial differential equation solution. Next, we take the one-dimensional Bratu problem as an example and use the finite difference method to discretize it.

For the one-dimensional Bratu problem (1.1), due to its simpler form, we can discretize the differential equation through the second-order difference formula of the second-order derivative function.

We first divide the interval $[0, 1]$ into N equal parts, which means that the length of the grid unit is $h = \frac{1}{N}$, and the points on the grid are $x_i = ih, i = 0, 1, 2, \cdots, N$, and from the boundary conditions we can get $x_0 = x_N = 0$.

Through the second-order difference formula of the second-order derivative function, we can get the discrete algebraic equations, as shown in the formula (2.5),

$$\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{bmatrix} + C \begin{bmatrix} \exp(u_1) \\ \exp(u_2) \\ \vdots \\ \exp(u_{N-2}) \\ \exp(u_{N-1}) \end{bmatrix} = 0 \tag{2.5}$$

where $C > 0$ is a parameter of the system.

We can simply express the above nonlinear parameter-containing system as,

$$Au + N(C, u) = 0 \tag{2.6}$$

For Rayleigh-Bénard problem, the procedure is similar.

2.2 Pseudo-arclength continuation

In this section, we introduce the Pseudo-arclength continuation method[5] for solving nonlinear systems with parameters. For nonlinear parameter-containing systems, the simplest algorithm is the natural continuation algorithm, but for nonlinear parameter-containing systems with turning points, the natural continuation algorithm will often fail. It is because the natural continuation algorithm will iterate to the area where the solution does not exist, which leads to the failure of the algorithm[2]. In this case, the Pseudo-arclength continuation needs to be used.

Assume the nonlinear parameter-containing system to be solved is

$$\Phi(x, \mu) = 0$$

where μ is the system parameter. Our goal is to get the solution set $\{(x, \mu)\}$ of the system.

Pseudo-arclength continuation method is an effective algorithm for solving the system. The solution set $\{(x, \mu)\}$ represents a curve, and the point (x, μ) in the solution set represents a point on the curve. We let it be described by the arc length parameter s, That is $(x(s), \mu(s))$.

At this time, the curve represented by the solution set is expressed as $r(s) = (x(s), (s))$. The curve needs to satisfy a regularization constraint:

$$\zeta \left\| \frac{dx}{ds} \right\|_2^2 + (1-\zeta)(\frac{d\mu}{ds})^2 = 1$$

where $\zeta \in (0, 1)$.

The Pseudo-arclength continuation method starts from an initial solution and iteratively obtains all the solutions step by step. The algorithm is divided into two parts. The first part is called predition, which obtains the predicted value of the next solution, and the second part is called correction, which corrects the predicted value.

In the first part, the predition part, we use the previously calculated points or initial values to calculate the predicted value of the next point on the curve, namely

$$(x^0, \mu^0) = (x_i, u_i) + \Delta s(\dot{x}, \dot{\mu})$$

where (x_i, u_i) is the previously obtained solution or initial value, (x^0, μ^0) the predicted value of the next point, and Δs an appropriate step size.

One can use the formula (2.7) to calculate $(\dot{x}, \dot{\mu})$,

$$(\dot{x}, \dot{\mu}) = \frac{(x_i, \mu_i) - (x_{i-1}, \mu_{i-1})}{s_i - s_{i-1}} \tag{2.7}$$

However, it is worth noting that when we use the initial value to solve the second solution, there is not enough data to calculate $(\dot{x}, \dot{\mu})$. In this case, one can use the derivative of $F(x, \mu) = 0$ to get the initial $(\dot{x}, \dot{\mu})$.

We first take the derivative of the nonlinear system with respect to μ, and get

$$\frac{\mathrm{d}}{\mathrm{d}\mu}\Phi(x, \mu) = \Phi_x \frac{\partial x}{\partial \mu} + \Phi_\mu = 0 \tag{2.8}$$

Next, we have

$$\frac{\partial x}{\partial \mu} = \frac{\partial x}{\partial s}\frac{\partial s}{\partial \mu} \tag{2.9}$$

Bringing it into the regularisation constraint, we get

$$\frac{\partial \mu}{\partial s} = (1 + \zeta \left\| \frac{\partial x}{\partial \mu} \right\|_2^2)^{-\frac{1}{2}} \tag{2.10}$$

Through formula(2.8)(2.9)(2.10), we can get $(\dot{x}, \dot{\mu})$.

In the second part, that is, in the correction, we correct the predicted value obtained in the first part to make it as close to the exact solution as possible. In this part, we need to solve $\Phi(x, \mu) = 0$ and a discretized regularisation constraint

$$r(x, \mu) = \Delta s^2 - \zeta(x - x_i)^T(x - x_i)) - (1-\zeta)(\mu - \mu_i)^2 = 0 \tag{2.11}$$

Finally, a linear system of the form

$$\begin{bmatrix} \Phi_x(x^k,\mu^k) & \Phi_\mu(x^k,\mu^k) \\ r_x(x^k) & r_\mu(\mu^k) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\mu \end{bmatrix} = \begin{bmatrix} -\Phi(x^k,\mu^k) \\ -r(x^k,\mu^k) \end{bmatrix} \tag{2.12}$$

needs to be solved.

After solving the linear system (2.12) to get $\Delta x, \Delta\mu$, using

$$x^{k+1} = x^k + \Delta x$$
$$\mu^{k+1} = \mu^k + \Delta\mu$$

we can correct the predicted value until it meets our accuracy requirements.

In addition, we can also obtain Δx and $\Delta\mu$ by solving a linear system (2.13):

$$\begin{cases} \Phi_x y &= \Phi_\mu \\ \Phi_x z &= -\Phi \end{cases} \tag{2.13}$$

Assuming that Φ_x is reversible, the first equation of the linear system in the formula (2.12) becomes

$$\Delta x + y\Delta\mu = z \tag{2.14}$$

Putting it into the second equation of (2.12), we get

$$\Delta\mu = \frac{r_x z - r}{r_\mu - r_x y} \tag{2.15}$$

By formula (2.14) and formula (2.15) we can get $\Delta\mu$ and Δx.

In addition, we need to set convergence conditions to terminate the iteration. Usually, given the threshold ϵ, when

$$\left\|(\Delta x^T, \Delta\mu)\right\|_\infty < \epsilon \tag{2.16}$$

At this time, the correction part converges and stops continuing to iterate.

In the Pseudo-arclength continuation method, the selection of the step size Δs is very important, and this value determines whether the value obtained in the prediction part is accurate. If the value of Δs is too large, it will cause incorrect results of the algorithm or more iterations. If Δs is too small, it will cause too much calculation and the calculation time is too long, so we need to try our best choosing an appropriate step size that allows the algorithm to reduce the calculation time while maintaining the correctness of the results.

In addition, the Pseudo-arclength continuation algorithm involves the solution of linear systems. One can use direct solver or iterative method (such as GMRES algorithm[6]) to solve it.

2.3 Preconditioner

In this section, we introduce the preconditioning technique, and a widely used preconditioner, the ILU preconditioner.

In practical applications, the linear systems we need to solve are often large and sparse. At this time, the direct solver is often not applicable. Usually we use iterative algorithms to solve such linear systems.

However, iterative algorithms also have some shortcomings, such as slow convergence speed, lack of robustness, and so on. Preconditioning is an effective way to solve these problems.

Assume that we want to solve a linear system of the form

$$Ax = b \tag{2.17}$$

When the linear system is not easy to solve, we can convert it into a linear system with the same solution as the original system but easier to solve (with smaller condition numbers) through preconditioning.

Generally, there are three ways to apply preconditioning techniques. Regard the preconditioner as M. The first method multiplies the preconditioner on the left side of the linear system, namely

$$M^{-1}Ax = M^{-1}b \tag{2.18}$$

The second method is to load the preconditioner on the right side of the linear system, that is

$$AM^{-1}u = b \tag{2.19}$$

Among them, $x = M^{-1}u$.

The third first get the decomposition of the preconditioner

$$M = M_L M_R \tag{2.20}$$

Then apply it to the linear system, that is

$$M_L^{-1}AM_R^{-1}u = M_L^{-1}b \tag{2.21}$$

where $x = M_R^{-1}u$.

Normally, we have two requirements for the preconditioner we used:

1. The preconditioner should be close to the coefficient matrix A and not singular
2. The condition number of the new linear system obtained after applying the preconditioner is smaller and easier to solve.

In this thesis, we use a widely applicable preconditioner called ILU preconditioner.

2.3.1 ILU preconditioner

In numerical linear algebra, the incomplete LU decomposition of the coefficient matrix A as a preconditioner has proved to be a widely effective method[7].

First, the definition of incomplete LU decomposition is given:

Given a square matrix A of order n, we can define graph $G(A)$ as

$$G(A) = (i, j) \in \mathbb{N}^2 : A_{ij} \neq 0$$

With the help of $G(A)$, we can define the sparse mode S, which satisfies

$$S \subset 1, ..., n^2, \quad (i,i) : 1 \leq i \leq n \subset S, \quad G(A) \subset S$$

We call decomposition $A = LU - R$ as incomplete LU decomposition, if it satisfies:

1. $L \in \mathbb{R}^{n \times n}$ is a lower triangular matrix
2. $R \in \mathbb{R}^{n \times n}$ is an upper triangular matrix The items of
3. L, U outside the sparse mode are all 0, that is, $L_{ij} = U_{ij} = 0, \forall (i,j) \notin S$
4. $R \in \mathbb{R}^{n \times n}$ The item in the sparse mode is 0, that is, $R_{ij} = 0, \forall (i,j) \in S$

We take the LU in the incomplete LU decomposition as our preconditioner, that is, $M = LU$. This type of preconditioner is a widely applicable preconditioner, and it has proved to be effective[7].

2.4 HYMLS

HYMLS[8] is a hybrid direct/iterative solver for the Jacobian of the incompressible Navier-Stokes equations on structured grids.

For Rayleigh-Bénard problem

- we use HYMLS to create the ILU preconditioner.
- we use the solver in the HYMLS to solve Rayleigh-Bénard problem.

3 One-dimenstional Bratu problem

In this chapter, pseudo-arclength continuation is used to solve one-dimensional Bratu problem numerically. Also, we study the singularity occurring during bifurcation analysis and research the relationship between several parameters and the results.

3.1 Effect of methods

In this section, we compare the time taken when we use different algorithms to solve the linear systems involved in the Pseudo-arclength continuation method.

In real life, the linear systems we encounter are often large and sparse. It is often a better choice to use iterative algorithms to solve this type of linear system. This is because there is an accumulation of errors in the direct solver. When the system is large, the error will also expand, which will affect the correctness of our solution. It is worth noting that although the error of the iterative algorithm is smaller, the running speed of the iterative algorithm has no advantage compared with the direct algorithm. Even the iterative algorithm often encounters problems such as slow convergence speed. At this time, we usually use preconditioning techniques to accelerate Iterative algorithm.

Here, we set the number of grid cells N to 128, $\Delta s = 0.1$, and then use three different methods to solve the linear system (2.12) and the linear system (2.13), respectively. The first method is the direct solver. We use LU decomposition and forward substitution algorithm. The second method is the GMRES method without preconditioning. Here, we set $restart = 20$, that is, the number of iterations between restarts, and $tolerance = 1e-05$. The third method is the GMRES method using ILU preconditioner. We call it as pre-GMRES. Here, we use the *spilu* function from the *linalg* package in python to create the ILU preconditioner. We also set $restart = 20$ and $tolerance = 1e-05$. The results are shown in the Table 3.1:

Table 3.1: The algorithm running time(unit:second)

N	Linear system2.12			Linear system2.13		
	Direct solver	GMRES	pre-GMRES	Direct solver	GMRES	pre-GMRES
64	0.007	10.414	0.156	0.012	12.406	0.290
128	0.012	33.299	0.221	0.019	68.946	0.402
512	0.039	277.251	0.576	0.058	1170.760	1.000
1024	0.063	971.963	0.777	0.119	7762.917	1.366

From the results of the Table 3.1, we can see that when the dimension of the linear system is low, whether it is solving the linear system (2.12) or the linear system (2.13), the direct solver is much faster than the iterative method. However, it is worth noting that when the dimension of the linear system continues to increase, the time spent in the direct solver increases relatively faster than the preconditioned iterative method.

In addition, it can be seen that the preconditioned GMRES algorithm converges much faster than the GMRES algorithm without preconditioning. This shows that the ILU preconditioner we use is very effective. It greatly accelerates the convergence of the algorithm.

In the following numerical experiments, due to the extremely low efficiency of the unpreconditioned GMRES algorithm, we abandon the use of this algorithm and only compare the direct solver with the preconditioned GMRES algorithm.

At the same time, we also notice that under the same conditions, it takes less time to solve the linear system (2.12). Although the linear system (2.12) is more complicated, the solution of the linear system (2.13) requires the solution of two linear systems, resulting in the actual time required to solve the problem.

Moreover, we can see that for the direct solver and the preconditioned GMRES algorithm, if we double the number of grid cells, the algorithm running time is also almost doubled, which indicates that the algorithm time is $\mathcal{O}(N)$.

In summary, when the dimension of the linear system is low, the direct solver is a faster and more effective method. However, errors will accumulate in the process of direct solver, and the influence of errors is often not obvious in low-dimensional problems, but when the dimension is large, the deviation of the solution results may be too large. Therefore, we need to select an appropriate method according to the actual situation encountered. When the problem dimension is small, the direct solver can be used to obtain a satisfactory answer, but when the problem dimension is large, the iterative method is a better choice.

3.2 Effect of the number of cells

In this section, we first set $\Delta s = 0.1$. Then, we change the number of cells and observe its influence on the results.

We know that for most differential equations, we often cannot get analytical solutions, we can only use numerical method to get numerical solutions. In the process of solving, we need to discretize the differential equation. In this process, the more discrete points we choose, the closer the numerical solution should be to the exact solution.

Here, we use two different methods to solve the one-dimensional Bratu problem, respectively. First, we use the direct solver and preconditioned GMRES algorithm to solve the linear system (2.12), respectively. Then, we use the direct solver and preconditioned GMRES algorithm to solve the linear system (2.13), respectively. For preconditioned GMRES algorithm, we also set $restart = 20$ and $tolerance = 1e-05$. The results are shown in Figure 3.1 and Table 3.1:

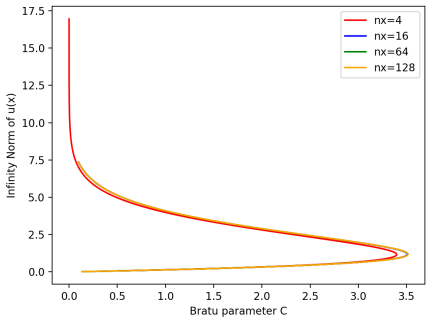

Figure 3.1: Effect of the number of cells on the critical parameter value

From Figure 3.1 we can see that the higher the number of grid cells, the closer our results are to the exact solution $C_c = 3.513830719$.

Below we list the bifurcation point obtained by each method, that is, the maximum value of the parameter C we obtained, as shown in the Table 3.2:

Table 3.2: Effect of the number of cells on the turing point we get

N	Linear system(2.12)		Linear system(2.13)	
	Direct solver	preconditioned GMRES	Direct solver	preconditioned GMRES
4	3.3955425693283297	3.395542718391918	3.3955425693283297	3.395542569270278
16	3.5065373528713466	3.5065373138008047	3.506537352871346	3.506537353003978
64	3.5132934860734037	3.513293501931244	3.5132934860733984	3.513293486073435
128	3.5136815558555115	3.513681556047259	3.51368155585553	3.51368155585596

From the Figure 3.1 and the Table 3.2, we can easily see that in the same grid, using different methods to solve the problem, the results obtained are very close, which means that the accuracy of each algorithm is basically the same. However, it's worth noting that as we get closer and closer to the true turning point, the linear system (2.13) will also get closer and closer to the singularity. At this time, the algorithms applied to the linear system (2.13 may fail. However, in our experiments, we have not encountered any cases where the algorithm fails, so we think that the algorithm does not fail in general. We guess that it will fail only in the case of extremely high accuracy.

Another observation is that when we continue to densify the grid, that is, the number of grid cells is getting larger, no matter which algorithm is used to solve the problem, the maximum value of C we get is getting closer and closer to the exact value $C_c = 3.513830719$. This is consistent with our theory, since the denser the grid, the closer the obtained numerical solution is to the real solution. However, the denser the grid, the greater the amount of calculation and the greater the calculation time. In actual situation, we often need to find a balance between accuracy and calculation time.

In summary, in this part, we verified that the larger the number of the grid cells, the closer our result is to the exact solution.

3.3 Effect of Δs

In this section, we first set the number of grid cells $N = 128$. Then we change the value of Δs in the Pseudo-arclength continuation method and observe the effect of the change of this variable on the result.

According to the theory, the smaller the iteration step Δs, the closer the solution obtained in the prediction part of the Pseudo-arclength continuation method is to the exact solution, but the corresponding calculation time we need is longer.

Here, we also use two different methods to solve the one-dimensional Bratu problem, respectively. First, we use the direct solver and preconditioned GMRES algorithm to solve the linear system (2.12), respectively. Then, we use the direct solver and preconditioned

GMRES algorithm to solve the linear system (2.13), respectively. For preconditioned GMRES algorithm, we also $restart = 20$ and $tolerance = 1e-05$.

Below we list the turning points obtained by each method under different Δs, that is, the maximum value of the parameter C obtained. The results are shown in the Table 3.3:

Table 3.3: Effect of Δs on the results

Δs	Linear system2.12		Linear system2.13	
	Direct solver	pre-GMRES	Direct solver	pre-GMRES
1	3.509575114557196	3.509575101442395	3.5095751145571983	3.5095751228541534
0.1	3.5136815558555115	3.513681556047259	3.51368155585553	3.51368155585596
0.01	3.5137187391364377	3.513765004259654	3.5137187391364555	3.5137619069145747
0.001	3.513719135866797	3.513758776919808	3.5137191358667965	3.513759181816871

From the results in the Table 3.3, we can see that when Δs keeps getting smaller, the numerical solution obtained under the same grid is more accurate, that is, closer to the exact solution $C_c = 3.513870319$. When Δs changes from 0.1 to 0.01, the accuracy is greatly improved. This is because the smaller the Δs, the more accurate the results obtained in the prediction part of the Pseudo-arclength continuation method. However, when Δs changes from 0.01 to 0.001, the accuracy increase is small, but in this case, our calculation amount and calculation time will increase a lot, that is to say that it is not a reasonable choice for $\Delta s$to change from 0.01 to 0.001, and its benefits are not high.

It is worth noting that the smaller the Δs, the closer we are to the true turning point. In this case, the coefficient matrix of our linear system (2.13) is also closer to the singularity, and the algorithm may fail.

In summary, through a number of numerical experiments, we have verified the influence of the iteration step size Δs on the results, that is, when the iteration step size Δs is smaller, the numerical solution we get is closer to the exact solution. Combining the above numerical experiments, we can think that $N = 128, \Delta s = 0.01$ is a set of reasonable parameter choices, and their combination can make the algorithm get an accurate numerical solution in a short time.

4 Rayleigh-Bénard problem

In this section, we take the Rayleigh-Bénard problem as an example to explore the linear stability of the solution.

4.1 Effect of the number of Levels

In this section, we observe the effect of the change of the number of levels on the results under different targets.

Before we say something about the numerical results, we need to explain what we mean by the number of levels. When we solve the correction equation which is a part of JDQZ method, we use multilevel preconditioner to speed up the convergence[8]. If we set the number

of levels to 2, that means that we are using two-level ILU preconditioner. Hence, the number of levels in our experiments represents the number of level of ILU preconditioner.

First, we set the number of grid cells N to 32 and the Rayleigh number to 1553.083383, which means that we are really close to the real bifurcation point. The experimental results are shown in the Table 4.1 and Table 4.2,

Table 4.1: Effect of the levels of preconditioner on the number of iterations of JDQZ and GMRES when Rayleigh number is close to the bifurcation point

The number of levels	The number of iterations of JDQZ		The total number of iterations of GMRES	
	equation 1.10	equation 1.9	equation 1.10	equation 1.9
0	16	16	58	158
1	51	failed	5043	failed

From the Table 4.1, one can see that the number of levels has a great influence on the number of iterations of JDQZ and GMRES methods when we are close to the bifurcation point. The smaller the number of levels, the faster the algorithm converges. Note that when the number of levels is 1, the algorithm always fail when solving Equation 1.10. In very few cases it converges, but the number of iterations of JDQZ and GMRES method are large, which indicates that this is bad convergence. This is because we are really close to the bifurcation point. In this case, Equation 1.10 is close to the singular system, which may lead to the failure of the algorithm.

In the following, we list the calculated eigenvalues, and the results are shown in Table 4.2 and Table 4.3.

Table 4.2: Eigenvalues when we are close to the bifurcation point(Solving Equation 1.10)

number	The number of levels			
	0		1	
	α	β	α	β
1	3.14552647e-09	0.00120876	3.46476361e-09	0.00129467
2	-3.09458868e-04	0.00099912	-3.09702198e-04	0.00102635
3	-4.78357326e-05	0.00174102	-4.52708087e-05	0.00163703
4	-7.43945212e-04	0.00193454	-7.41916785e-04	0.00186894
5	-8.58971795e-04	0.00102366	-8.45711788e-04	0.00102972

Table 4.3: Eigenvalues when we are close to the bifurcation point(Solving Equation 1.9)

number	The number of levels	
	0	
	α	β
1	3.20445504e-09	0.00121679
2	-3.07165224e-04	0.00101014
3	-4.81135293e-05	0.00174101
4	-7.42947878e-04	0.00189302
5	-8.48508134e-04	0.00100707

From the Table 4.2 and Table 4.3, we can see that there is no obvious difference between eigenvalues obtained by the two methods. That means that when the algorithm is able to converge successfully, the choice of method and the number of levels do not have a significant effect on the eigenvalue results.

Subsequently, we set the number of grid cells N to 32 and the Rayleigh number to 10, which means that we are far away from the bifurcation point. The experimental results are shown in the Table **??**,

Table 4.4: Effect of the levels of preconditioner on the number of iterations of JDQZ and GMRES when Rayleigh number is far away the bifurcation point

The number of levels	The number of iterations of JDQZ		The total number of iterations of GMRES	
	equation 1.10	equation 1.9	equation 1.10	equation 1.9
0	23	22	78	110
1	24	23	774	810

From the Table 4.4, one can see that The number of levels has the same effect on the results as before when we are far away from the bifurcation point.

We also list the calculated eigenvalues, and the results are shown in Table 4.5 and Table 4.6.

Table 4.5: Eigenvalues when we are far away from the bifurcation point(Solving Equation 1.10)

number	The number of levels			
	0		1	
	α	β	α	β
1	-0.01467388	0.00120876	-0.01457547	0.0034663
2	-0.01718342	0.00418236	-0.01716208	0.00417716
3	-0.02071538	0.00461037	-0.02080591	0.00463052
4	-0.02569372	0.00516867	-0.02567186	0.00516427
5	-0.0313049	0.00555606	-0.03131926	0.00555861

Table 4.6: Eigenvalues rb3 when we far away from the bifurcation point(Solving Equation 1.9)

number	The number of levels			
	0		1	
	α	β	α	β
1	-0.01466233	0.00348695	-0.01458602	0.0034688
2	-0.01718085	0.00418173	-0.01714411	0.00417279
3	-0.02072677	0.0046129	-0.02077923	0.00462458
4	-0.02569154	0.00516823	-0.0256744	0.00516478
5	-0.03130646	0.00555633	-0.03130038	0.00555525

From the Table 4.5 and Table 4.6, we can see that the experimental results are similar with the first case. That means that the number of levels has similar effects on the results, and has nothing to do with whether the point is a bifurcation point.

4.2 Effect of the Rayleigh number

In this section, we set the grid $(32 \times 16 \times 1)$, Δs to 600, the number of levels to 1, and observe the influence of the change of the Rayleigh number on the result. The experimental results are shown in the Table 4.7, Table 4.9 and Table 4.8,

Table 4.7: Effect of Rayleigh number on the number of iterations of JDQZ and GMRES when solving Equation 1.9

Rayleigh number	The number of iterations of JDQZ	The total number of iterations of GMRES
1555	29	8091
1554	57	17672
1553.16	76	25871
1553.15	failed	failed
1553.1	failed	failed
1553.083383	failed	failed
1553.01	failed	failed
1553	19	3418
1552	28	7401

Table 4.8: Effect of Rayleigh number on the number of iterations of JDQZ and GMRES when solving Equation 1.10

Rayleigh number	The number of iterations of JDQZ	The total number of iterations of GMRES
1555	19	4074
1554	19	4053
1553.16	24	6229
1553.15	18	4091
1553.1	16	2963
1553.083383	50	14069
1553.01	28	7112
1553	18	3481
1552	18	3515

Here, we specify that if the algorithm does not converge, or if the results of multiple calculations are inconsistent, we consider the algorithm to have failed. From the results in the Table 4.7, we can see that the algorithm can easily fail if the Rayleigh number is in the range of 1553.01 to 1553.15. In other ranges, the algorithm converges successfully and the results are consistent across multiple calculations.

In addition, we can see that solving Equation 1.10 requires fewer iterations of the JDQZ and GMRES methods than solving Equation 1.9, and is not affected by singularity.

We are also interested in the difference between the two methods as we gradually move away from the bifurcation point. In Table 4.9, we list the first eigenvalues (α, β)calculated by the two methods for different Rayleigh numbers.

Table 4.9: Eigenvalues calculated by the two methods for different Rayleigh numbers.

target	equation 1.10	equation 1.9
1553.083383	(-3.09458868e-04, 0.00102366)	failed
1553	(-4.21546129e-07, 0.00120669)	(-4.34629297e-07, 0.00123195)
1552	(-5.58146056e-06, 0.00121147)	(-5.54714796e-06,0.00120185)
1551	(-1.07674477e-05,.00121229)	(-1.08714550e-05,0.00122428)
1550	(-1.59444019e-05, 0.00121266)	(–1.63625136e-05, 0.00124464)
1250	(-0.00165097, 0.0012432)	(-0.00165218, 0.00124412)
1000	(-0.00243738,0.00101597)	(-0.00242862, 0.00101232)

From the above experimental results in the Table 4.9, we can see that the smaller the Rayleigh number, the closer the eigenvalues obtained by the two methods are. This is because the smaller the Rayleigh number is, the further away we are from the bifurcation point. this means that solving Equation 1.9 is further away from the singularity, making the algorithm converge better and the solution results more accurate.

5 Discussion and Conclusions

At the beginning of the research, We first briefly introduce the singular systems that arise when solving the one dimensional Bratu problem and the Rayleigh Bénard problem, and introduce two methods for solving the problem. Then, we introduce several algorithms we use in detail, such as finite difference method, pseudo-arclength continuation method and preconditioning technique. Finally, these theories are applied to one-dimensional Bratu problem and Rayleigh-Bénard problem.

Next, a number of numerical results are introduced. We first solve these two problems numerically, then change parameters in the system to see the effect of these parameters on the numerical results.

The experimental results shows that, for one-dimensional Bratu problem, (i)when the iteration step size Δs is smaller, the numerical solution we get is closer to the exact solution;(ii) The larger the number of the grid cells, the closer our result is to the exact solutionCombining the above numerical experiments, we can say that $N = 128, \Delta s = 0.01$ is a set of reasonable parameter choices, and their combination can make the algorithm get a accurate numerical solution in a short time. (iii)In addition, we found that it is usually difficult for the algorithm to fail when we are close to the bifurcation point.

For Rayleigh-Bénard problem, the experimental results shows that(i)the smaller the Rayleigh number, the closer the results of the two methods; (ii)the larger the level of the ILU preconditioner, the less precise the numerical solution obtained and the greater the number of iterations of JDQZ and GMRES methods, and the more the calculation time. (iii) In addition, we found that the algorithm can easily fail if the Rayleigh number is in the range of 1553.01 to 1553.15, which is close to the bifurcation point. In other ranges, the algorithm converges successfully and the results are consistent across multiple calculations.

Finally, the research of this project provides direction guidance and method innovation for the research of related topics, and there is still a lot of work to be done to quantify the more specific characteristics.

References

[1] Adel Mohsen. A simple solution of the bratu problem. *Computers & Mathematics with Applications*, 67(1):26–33, 2014.

[2] Weiyan Song. *Matrix-based techniques for (flow-) transition studies*. PhD thesis, University of Groningen, 2019.

[3] Diederik R Fokkema, Gerard LG Sleijpen, and Henk A Van der Vorst. Jacobi–davidson style qr and qz algorithms for the reduction of matrix pencils. *SIAM journal on scientific computing*, 20(1):94–125, 1998.

[4] Fred Wubs. Lecture notes:computational method of science. 2021.

[5] Sven Baars. *Numerical methods for studying transition probabilities in stochastic ocean-climate models*. PhD thesis, Rijksuniversiteit Groningen, 2019.

[6] Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[7] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[8] Sven Baars, Mark van Der Klok, Jonas Thies, and Fred W Wubs. A staggered-grid multilevel incomplete lu for steady incompressible flows. *International Journal for Numerical Methods in Fluids*, 93(4):909–926, 2021.

[9] Henk A Dijkstra, Fred W Wubs, Andrew K Cliffe, Eusebius Doedel, Ioana F Dragomirescu, Bruno Eckhardt, Alexander Yu Gelfgat, Andrew L Hazel, Valerio Lucarini, Andy G Salinger, et al. Numerical bifurcation methods and their application to fluid dynamics: analysis beyond simulation. *Communications in Computational Physics*, 15(1):1–45, 2014.

[10] Thomas Erik Mulder Mulder. *Design and bifurcation analysis of implicit Earth System Models*. PhD thesis, Utrecht University, 2019.

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