Computing Resonant States of a Quantum Mechanical Three-body Problem on Supercomputers

Zhaonan Meng

Delft University of Technology

TU Delft

Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

Computing Resonant States of a Quantum Mechanical Three-body Problem on Supercomputers

A thesis submitted to the
Delft Institute of Applied Mathematics
in partial fulfillment of the requirement

for the degree

MASTER OF SCIENCE
in
APPLIED MATHEMATICS

by

Zhaonan Meng

Delft, The Netherlands
August 16, 2023

Computing Resonant States of a Quantum Mechanical Three-body Problem on Supercomputers

by

Zhaonan Meng

Student Number:	5841003
Project Duration:	September, 2022 - August, 2023
Faculty:	Faculty of Electrical Engineering, Mathematics and Computer Science, Delft
Thesis Committee:	Prof. Dr. Martin van Gijzen, TU Delft (Chair of committee)
	Dr. Jonas Thies, TU Delft (Daily supervisor)
	Dr. Johan L.A. Dubbeldam, TU Delft

TU Delft

Preface

This thesis paper showcases the work I have conducted in my thesis project during my Master's program in Computer Simulations for Science and Engineering (COSSE). COSSE is a dual-degree program organized jointly by KTH Royal Institute of Technology Stockholm (Sweden), Technical University of Berlin (Germany), and Delft University of Technology (the Netherlands) with KTH acting as the coordinating institution of the Consortium. I completed my first year of studies at TU Berlin and subsequently conducted this thesis project at TU Delft, under the supervision of Prof. Martin van Gijzen and Dr. Jonas Thies.

The overall thesis project consists of two parts: a thorough literature review and the actual thesis work. The literature review took place between September 2022 and March 2023, during which I investigated the physical background and related methodology. Chapter 1, 2 and 3 comprehensively cover the contents of the literature review. Following that, the thesis delves into the implementation and results, providing answers to the research questions posed.

This thesis project covers my research interests in numerical linear algebra and high-performance computing. The research project was conducted in collaboration with German Aerospace Center (DLR), which provided me with support on quantum physics. Through the project, I learned how to conduct research on my own and acquired in-depth knowledge and expertise in cutting-edge technology in eigenvalue problems and high-performance computing.

I would like to express my heartfelt gratitude to Martin van Gijzen and Jonas Thies for their invaluable instructions and support throughout the entire process of crafting my thesis project. Their expertise and insightful feedback has been instrumental in shaping the quality and depth of my research. Thanks to their guidance, I have gained the knowledge required to conduct research.

I am also grateful to Maxim Efremov and Matthias Zimmermann from German Aerospace Center for imparting their knowledge of physics, which contributes to the background and application parts of my thesis. Furthermore, I would like to extend my sincere appreciation to Johan L.A. Dubbeldam for accepting the invitation to be a member of my graduation committee.

Lastly, I would like to thank my parents for their support and care during my time in Europe. It is always challenging to be far away from one's hometown, and it was my parents who gave me the courage to pursue my studies in Europe.

Zhaonan Meng
Delft, August 2023

Abstract

This thesis aims to develop an advanced numerical solver capable of efficiently computing the resonant states of quantum mechanical two-body and three-body problems, thereby expanding our understanding of these complex systems. The quantum three-body problems feature at least two dimensions, which necessitates substantial computational efforts. Therefore, in order to tackle these challenging computations, we need to seek assistance from supercomputers. By harnessing the capabilities of high-performance computing, we can significantly reduce the amount of time spent waiting for programs to run for hours.

In this thesis, we first introduce some basic knowledge about quantum few-body problems and resonant states, showing how the physical problem gives rise to a mathematical problem, the quadratic eigenvalue problem (QEP). Building upon the physical background, our journey in developing the methodology begins with two fundamental components: discretization and eigensolver. The pseudo-spectral methods are introduced to represent the quadratic eigenvalue problem as a matrix problem, by which we can solve the problem numerically through some eigensolvers. We describe a classical approach called linearization for solving QEPs, which transforms the quadratic problem into a generalized eigenvalue problem. Following the linear transformation, we apply the Jacobi-Davidson QZ (JDQZ) method, an iterative eigensolver, to solve the linearized problem. Alternatively, we could also use the Jacobi-Davidson (JD) method to approximate the quadratic eigenvalue problem's eigenpairs directly. In this thesis, we provide an outline of the Jacobi-Davidson process for solving both linear and quadratic problems. Two routes for solving QEPs are utilized and compared: linearization combined with the Jacobi-Davidson QZ method, and the quadratic Jacobi-Davidson method. Through our research and analysis, we demonstrate that the Jacobi-Davidson algorithm exhibits superior computational efficiency when adapted to solve QEPs directly.

Another significant objective of this thesis is to parallelize the eigensolver on supercomputers. We implement a hybrid distributed/shared memory parallelization of the Jacobi-Davidson algorithm to solve quadratic eigenvalue problems that arise from one-dimensional three-body problems. We leverage the tensor structure inherent in the three-body problem to optimize computational efficiency. Specifically, we implement an efficient tensor product scheme for the application of the stiffness and damping operators, which are realized as dense matrix-matrix products. By incorporating a preconditioner that also preserves the tensor structure, we enhance the performance of our Jacobi-Davidson algorithm in computing three-body resonance poles within an acceptable speed.

Contents

List of Figures

List of Algorithms

List of Tables

Nomenclature

Abbreviations

Abbreviation	Definition
1D	One-dimensional
2D	Two-dimensional
AR	Arnoldi Method
BLAS	Basic Linear Algebra Subprograms
CAP	Complex Absorbing Potential
FDM	Finite Difference Method
FEM	Finite Element Method
GMRES	Generalized Minimal Residual Method
GS	Gram–Schmidt Process
JD	Jacobi-Davidson Method
JDQR	Jacobi-Davidson QR Method
JDQZ	Jacobi-Davidson QZ Method
LAPACK	Linear Algebra Package
MPI	Message Passing Interface
OpenMP	Open Multi-Processing
QEP	Quadratic Eigenvalue Problem
SEP	Standard Eigenvalue Problem
SPMD	Single Program, Multiple Data

Symbols

Symbol	Definition
$\vec{x}, \vec{y}$	Jacobi relative coordinates
V	Potential
v_0	Potential depth
E	Energy
k	Wave number
ψ	Wave function
α	Mass ratio
$\alpha_{x/y}$	Positive coefficients of three-body problems
$[-L_x, L_x],\ [-L_y, L_y]$	Spatial domain
$N_x,\ N_y$	Number of grid points
D_N	Chebyshev Differentiation matrix
I	Identity matrix
O	Zero matrix
$K^{(2b/3b)}$	Stiffness matrix for two/three-body problems
$C^{(2b/3b)}$	Damping matrix for two/three-body problems
$M^{(2b/3b)}$	Mass matrix for two/three-body problems
T	Quadratic polynomial matrix
P	Preconditioner

1 Introduction

Quantum few-body problems, emerging as a captivating research field, offer us a unique insight into the interaction of quantum particles. This introductory chapter gives an overview of the research of solving quantum few-body problems and serves as the gateway to our exploration. The fundamental knowledge of quantum few-body physics and resonant states will be provided. Following the introduction to physics, we discuss some previous related work, encompassing some few-body systems which have been studied and the tools employed in solving them. Based on the examination of previous work, we can formulate several research questions that will guide our investigation throughout the entire thesis, to find answers to these questions.

1.1. Physical background

To begin with, we would like to first provide a brief introduction to the physical background including the basic model of quantum two-body and three-body systems, respectively. In addition, we will give a detailed definition of resonant states of quantum systems, shedding light on their fundamental properties and behavior.

1.1.1. Quantum few-body systems

A quantum few-body system refers to a system composed of several interacting particles. The particles can be atoms, electrons, or other subatomic particles, whose behavior is governed by quantum mechanics. The study of quantum mechanical few-body problems such as electron atomic collisions [5, 23] has been pursued for decades, cultivating various fields where few-body physics plays an important role. The Efimov effect [21], which describes the emergence of an infinite sequence of universal states of three bosonic particles with s-wave resonant pair interactions in three dimensions, is one of the examples in which the few-body problem is involved.

The motion of all particles in a few-body system is governed by the Schrödinger equation. In the research, we consider the two-body and three-body systems introduced by [14]. We first consider a two-body system composed of a heavy particle of mass M and a light particle of mass m, giving the stationary Schrödinger equation

$$[-\frac{1}{2}\Delta_{\vec{x}} + V(\vec{x})]\psi(\vec{x}) = E\psi(\vec{x}) \tag{1.1}$$

for the wave function $\psi(\vec{x})$ and two-particle energy E, where $\Delta_{\vec{x}}$ denotes the Laplace operator with respect to $\vec{x}$ and $V(\vec{x})$ stands for the interaction term. $\vec{x}$ denotes the relative coordinate of the light particle concerning the heavy one. $\vec{x}$ can be reduced to a scalar x for a one-dimensional system. Note that for convenience we omit the constant terms such as Planck's constant in the equation. The potential $V(\vec{x})$ is given by

$$V(\vec{x}) = -v_0 f(\vec{x}), \tag{1.2}$$

where v_0 and $f(\vec{x})$ represent the interaction potential's magnitude and shape, respectively. We assume that f is symmetric, $f(\vec{x}) = f(|\vec{x}|)$, and $v_0 > 0$.

The potential $V(\vec{x}) = -v_0 f(\vec{x})$ depends on different mechanical systems. In physics, most two-body potentials vanish either exponentially or polynomially as $x \to \infty$. We consider both cases and focus on the attractive potential of the Gaussian shape

$$f_G(x) = \exp(-x^2) \tag{1.3}$$

and potential whose shape

$$f_L(x) = \frac{1}{(1+x^2)^3} \tag{1.4}$$

is determined by the cube of a Lorentzian.

In a three-body system with an additional heavy particle of mass M, an extra coordinate is required to describe the spatial relationships between the particles. Similar to two-body problems, the mass-imbalanced three-body system can be described by the dimensionless form of the stationary Schrödinger equation

$$[-\frac{\alpha_x}{2}\Delta_{\vec{x}} - \frac{\alpha_y}{2}\Delta_{\vec{y}} + V(\vec{x},\vec{y})]\psi(\vec{x},\vec{y}) = E\psi(\vec{x},\vec{y}) \tag{1.5}$$

for the three-particle wave function $\psi = \psi(\vec{x},\vec{y})$ corresponding to the three-particle energy E. $\Delta_{\vec{x}}$ and $\Delta_{\vec{y}}$ denote the Laplace operator concerning the relative coordinate vectors $\vec{x}$ and $\vec{y}$ respectively. As illustrated in figure 1.1, $\vec{x}$ denotes the coordinate of the light particle m concerning the center of mass of the two heavy particles M. And $\vec{y}$ represents the relative coordinate between the two heavy particles.

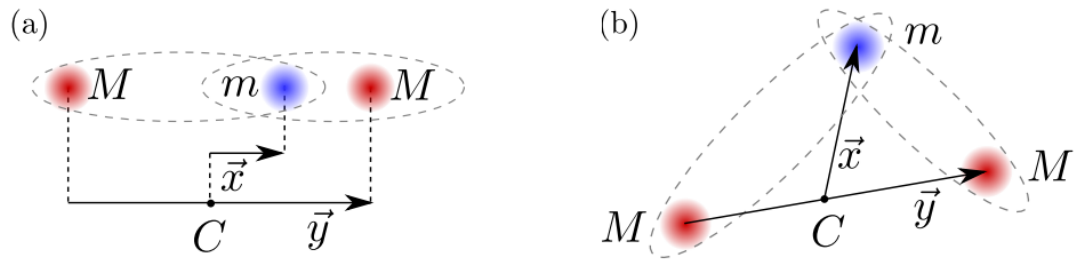


Figure 1.1: Three-body system in (a) one and (b) two spatial dimensions. Only interactions between heavy and light particles are considered. The center of mass of two heavy particles is denoted by C. Reproduced from Thies et al. (2022) [27].

The positive coefficients $\alpha_x = 2/(1+\alpha)$ and $\alpha_y = (1+2\alpha)/(2+2\alpha)$ are determined by the mass ratio $\alpha \equiv M/m$ of the heavy and light particle. When there is no interaction between heavy particles, the potential $V(\vec{x},\vec{y})$ can be divided into two terms

$$V(\vec{x},\vec{y}) = V(\vec{r}_+) + V(\vec{r}_-) = -v_0 f(|\vec{x} + \frac{1}{2}\vec{y}|) - v_0 f(|\vec{x} - \frac{1}{2}\vec{y}|), \tag{1.6}$$

where $|\vec{x} \pm \vec{y}/2|$ denotes the respective relative distance. The two terms model the interaction potential between the light particle and each heavy one.

By solving the stationary Schrödinger equation (1.1)/(1.5), which is in principle a standard eigenvalue problem (SEP), we can derive the wave functions (eigenvectors) and the corresponding observables (eigenvalues). The behavior of a few-body system is determined by the wave function, which describes the quantum state of the system. To study the behaviors of a few-body system more deeply, people compute different states of the system, such as bound states, anti-bound states, and resonant states. Section 1.2 will provide additional information regarding previous work on bound states, while our research primarily focuses on the study of resonant states. In the following subsection, we will show that to find resonant states one needs to impose some proper boundary conditions on the Schrödinger equation.

1.1.2. Resonant states

In this subsection, we begin the introduction to resonant states by presenting a quantum one-dimensional model. While numerous studies have focused on bound or scattering states within quantum one-dimensional models, there are also several works dedicated to exploring unstable quantum states, with which resonances can be identified [9]. Resonance is a phenomenon that manifests across a wide range of physics disciplines, spanning from classical mechanics to quantum mechanics. Meanwhile, the significance of the resonant phenomenon is growing, particularly in the realm of quantum mechanics concerning mesoscopic devices [16].

The resonant state of the open quantum system is studied from the viewpoint of the outgoing momentum flux [9, 16]. Before defining resonances, let us first consider a one-dimensional Schrödinger equation with compactly supported potentials on the real line

$$V(x) \in \mathbb{R},\ |V(x)| \leq C,\ V(x) = 0 \text{ for } |x| > L. \tag{1.7}$$

The solutions of the Schrödinger equation outside the potential are plane waves given by

$$\psi(x) = \begin{cases} Ae^{ikx} + Be^{-ikx} & \text{,for } x < -L, \\ Ce^{ikx} + De^{-ikx} & \text{,for } x > L, \end{cases} \tag{1.8}$$

where k denotes the wave number related to the energy E dimensionless by

$$E = \frac{1}{2}k^2. \tag{1.9}$$

We define the scattering matrix (S matrix) that relates the asymptotically incoming wave function with the asymptotically outgoing wave function [7]:

$$\begin{bmatrix} B \\ C \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} A \\ D \end{bmatrix}, \tag{1.10}$$

where $S_{11,12,21,22}$ are functions of the wave number k. The scattering matrix, denoted as the symbol S, is a fundamental concept in quantum physics. It serves as a crucial tool for explaining how particles or waves behave when they interact with and scatter off a potential. The S matrix connects the initial state of incoming particles with the final state of outgoing particles, providing information about the probabilities and amplitudes of different scattering events. The resonance is defined as a pole of the $S(k)$ matrix in the complex wave number plane. In complex analysis, the poles of a function are defined as follows:

Definition 1. *A point z_0 is called a zero of order m for the function $f(z)$ if f is analytic at z_0 and f and its first $m-1$ derivatives vanish at z_0, but $f^{(m)}(z_0) \neq 0$.*

Definition 2. *A point z_0 is called a pole of order m of $f(z)$ if $1/f$ has a zero of order m at z_0.*

Although in many textbooks, resonances are defined as poles of the S matrix, N. Hatano et al. [15, 16] shows that this definition is equivalent to solving the stationary Schrödinger equation under a certain boundary condition, known as the Siegert condition. Specifically, the Siegert condition is to have outgoing waves only and no incoming waves, i.e.,

$$\psi(x) = \begin{cases} Be^{-ikx} & \text{,for } x < -L, \\ Ce^{ikx} & \text{,for } x > L. \end{cases} \tag{1.11}$$

A square-well potential with the Siegert condition is illustrated in Fig. 1.2.

Therefore, we can establish a more convenient definition of the resonant state in the following way presented in [16]:

Definition 3. *A resonant state is a solution of the Schrödinger equation (an eigenfunction of the Hamiltonian) under the boundary conditions (1.11) that only the outgoing waves exist outside the potential cutoff $[-L, L]$.*

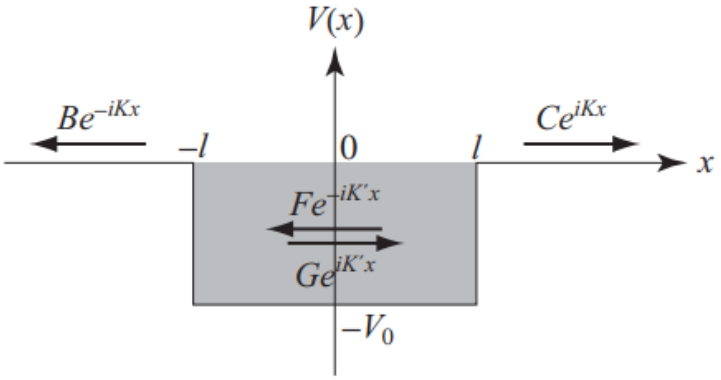


Figure 1.2: A square-well potential with the Siegert condition of outgoing waves only. Reproduced from N. Hatano et al. (2009) [15].

In other words, if the support of V is contained in $[-L, L]$, the resonance solutions can be computed by the stationary Schrödinger equation given by

$$[-\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + V(x) - \frac{1}{2}k^2]\psi(x) = 0 \text{ for } x \in [-L, L], \tag{1.12}$$

$$\begin{aligned}(\frac{\mathrm{d}}{\mathrm{d}x} + ik)\psi(x) = 0 \text{ at } x < -L,\\ (\frac{\mathrm{d}}{\mathrm{d}x} - ik)\psi(x) = 0 \text{ at } x > L.\end{aligned} \tag{1.13}$$

In our two/three-body problem, the potentials exhibit exponential or polynomial decay as $x \to \infty$. However, due to the impossibility of solving the problem over an infinite domain, it becomes necessary to introduce a sufficiently large cutoff on the domain length. This cutoff ensures that the potentials approach approximately zero at the boundary points. We denote L as the cutoff length. Therefore, the two-body Schrödinger equation holds within the interval $[-L, L]$, and the Siegert conditions hold at the two boundary points $x = -L$ and $x = L$. In conclusion, the two-body resonant problem is formulated as follows:

$$[-\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + V(x) - \frac{1}{2}k^2]\psi(x) = 0 \text{ for } x \in [-L, L], \text{ where } V(x) \approx 0 \text{ when } |x| = L. \tag{1.14}$$

$$\begin{aligned}(\frac{\mathrm{d}}{\mathrm{d}x} + ik)\psi(x) = 0 \text{ at } x = -L,\\ (\frac{\mathrm{d}}{\mathrm{d}x} - ik)\psi(x) = 0 \text{ at } x = L.\end{aligned} \tag{1.15}$$

Eq. (1.14) is a quadratic eigenvalue problem (QEP) in terms of k. A resonant state is a solution of the Schrödinger equation (1.14) under the boundary conditions (1.15) that only the outgoing waves exist outside the compact interval $[-L, L]$.

Fig. 1.3a and 1.3b illustrate the numerical solutions k and $E = k^2/2$ to Eqs. (1.14) with boundary conditions (1.15) for a square-well potential in $[-1, 1]$. The dots on the positive and negative imaginary k axis are the bound state and anti-bound state, respectively. The remaining dots represent resonant states, symmetrically lying on the lower half of the complex momentum plane with respect to the imaginary axis. In the energy plane, resonant states are located on the right half with respect to the imaginary E axis, while two (anti)bound states are both on the negative real E axis.

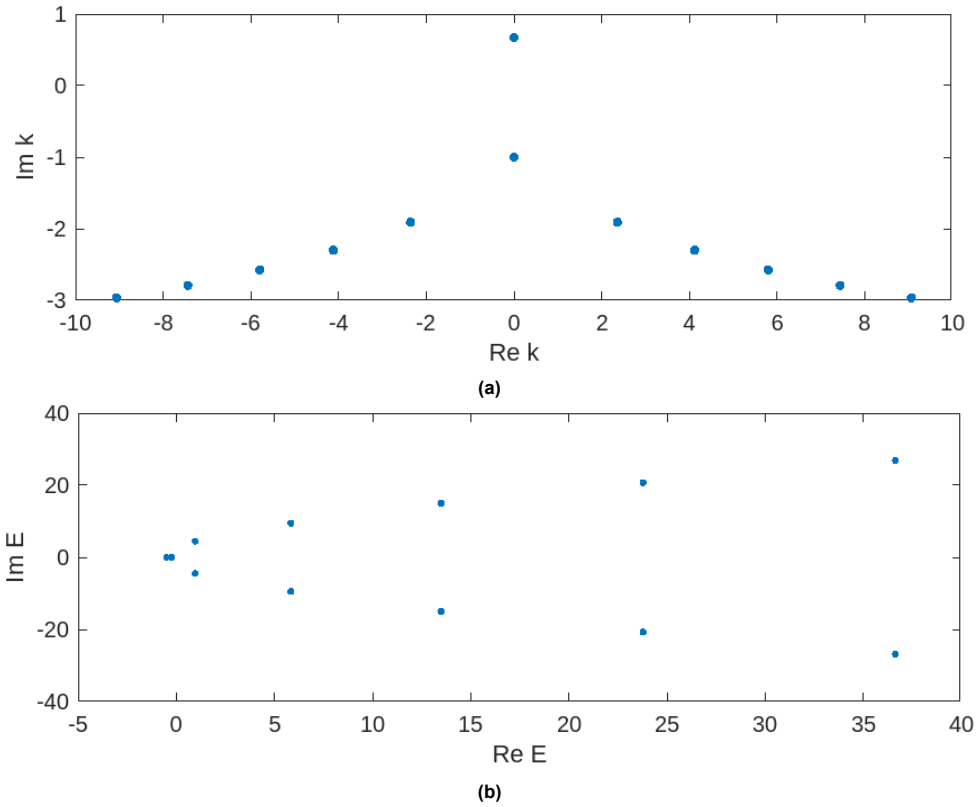


Figure 1.3: Solutions k and E for the square potential well ($V(x) = -1$ for $x \in [-1, 1]$, $V(x) = 0$ for $x \notin [-1, 1]$) with the Siegert condition.

Up to this point, the one-dimensional model we have discussed can be regarded as the simplest quantum two-body resonant model. After exploring two-body resonant systems, we can now shift our focus to the formulation of three-body problems, which involve the interaction of three quantum particles. Our research specifically focuses on one-dimensional three-body problems, where three quantum particles are confined to the same line. When considering a 1D three-body system, an additional coordinate (dimension) is required to describe the spatial relationships between the particles, as discussed in the introductory chapter. Therefore, the equations of the 1D three-body problem are given as follows:

$$[-\frac{\alpha_x}{2}\frac{\partial^2}{\partial x^2} - \frac{\alpha_y}{2}\frac{\partial^2}{\partial y^2} + V(x,y) - \frac{1}{2}k^2]\psi(x,y) = 0 \text{ for } (x,y) \in [-L_x, L_x] \times [-L_y, L_y], \tag{1.16}$$

$$\begin{aligned}(\frac{\partial}{\partial x} + ik)\psi(x,y) = 0 \text{ at } x = -L_x,\ (\frac{\partial}{\partial x} - ik)\psi(x,y) = 0 \text{ at } x = L_x,\\ (\frac{\partial}{\partial y} + ik)\psi(x,y) = 0 \text{ at } y = -L_y,\ (\frac{\partial}{\partial y} - ik)\psi(x,y) = 0 \text{ at } y = L_y.\end{aligned} \tag{1.17}$$

In the given equations, we use x and y to represent the coordinates between the particles instead of vector quantities $\vec{x}$ and $\vec{y}$. Since all the particles are located on the same line, using scalars is sufficient for describing the spatial relationship. Additionally, it is worth noting that $V(x, y)$ can be split into two parts: the potential between two heavy particles and the potential between the light particle and the center of mass of the two heavy particles. The cutoffs L_x and L_y should be chosen such that each potential component approximates zero at the respective boundaries. When there is no interaction, in other words, no potential, between the heavy particles, the boundary conditions are only applicable in the x direction.

In the present subsection, we have derived the equations to be solved in both the two-body case and the one-dimensional three-body case. This formulation suggests that the characterization of resonant states necessitates solving a quadratic eigenvalue problem. This realization serves as motivation to explore solvers designed to tackle QEPs. In the later methodology chapters, namely, Chapter 2 and

Chapter 3, we will demonstrate the necessary tools for discretizing and numerically solving the QEPs. Nevertheless, instead of jumping right into the methodology, we will begin by exploring the existing research pertaining to our topic in the following section.

1.2. Related work

Prior to commencing the actual investigation of the problem, it is essential to acquire a comprehensive understanding of some previous work in solving quantum few-body problems. As indicated in Section 1.1, people are currently trying to compute different states, such as bound states, anti-bound states, and resonant states, of the few-body systems. Several studies addressed to the analysis of bound states are reviewed in [18]. A bound state appears when a particle is subjected to a potential that restricts its movement, leading to localization in specific regions of space. In such states, the particle is confined by the potential, preventing it from freely dispersing across a wider spatial range. In contrast to bound states, the resonant state of the open quantum system is studied from the viewpoint of the outgoing momentum flux. The basic definition of the resonant state, as well as prior progress, are reviewed in [16] and [22], respectively. In many textbooks, resonance is defined as a pole of the S matrix. However, the article by N. Hatano et al. (2009) [15] states that it is more convenient to define a resonant state as an eigenstate of the Schrödinger equation with the Siegert condition, providing a probabilistic interpretation of resonant states.

Finding an analytic solution for bound or resonant states is usually impractical, except for some specific quantum systems. In the realm of bound states of few-body systems, Thies et al. (2022) [27] propose a computationally-efficient scheme for numerically computing bound states of two/three-body problems. A pseudo-spectral method based on the rational Chebyshev polynomials is applied to build a matrix representation of the Schrödinger equation. To solve the standard eigenvalue problem derived from discretization, several iterative eigensolvers for approximating eigenpairs of the Hamiltonian matrix, such as Krylov-Schur and the Jacobi-Davidson QR (JDQR) method with and without preconditioning, are tried and compared. To address the three-body problem which is more than one dimensional (1D), it is necessary to employ a Kronecker product structure when dealing with the associated Hamiltonian operator. In [27], a novel tensor method is presented to analyze the quantum mechanical three-body with local two-body interactions in 1D and 2D. The tensor method, referred to as the tensor product scheme, leverages the inherent structure of the Kronecker product to avoid storing redundant blocks. By exploiting the tensor product formulation, the method efficiently represents the Hamiltonian operator in higher dimensions while reducing the memory requirements. This approach allows for significant savings in terms of storage space, enabling more efficient computations for systems with large matrices. By developing a high-performance implementation of the tensor scheme which can be used on current supercomputers, [27] has shown for the first time the universal behavior of the 2D heavy-heavy-light three-body system when the ground-state energy of the heavy-light subsystems approaches zero. The method can be extended to studying resonant states of the three-body problems in 1D and 2D. In our research, we closely adhere to the fundamental concept presented in [27], aiming to identify resonant states by employing a similar methodology.

The field of numerically computing resonant states in quantum few-body systems is still relatively unexplored. As we will demonstrate in the following chapters, the accurate and efficient computation of resonant states poses unique challenges. Tolstikhin et al. (1998) [29] provide valuable inspiration for numerically computing resonant states, in the context of computing Siegert pseudostates defined as the outgoing wave solutions of the radial Schrödinger equation, specifically for systems with cut-off potentials. The resonant system is reduced to a quadratic eigenvalue problem (QEP) through the Galerkin discretization. To solve the derived quadratic eigenvalue problem, a conventional technique known as linearization is employed. This method involves transforming the quadratic eigenvalue problem into an equivalent linear eigenvalue problem, which can be more easily solved using existing linear eigenvalue solvers. Through [29], we can get a first impression that the resonant few-body systems can be reduced to an algebraic quadratic eigenvalue problem which can be solved by linearization. However, linearization is not the only way to solve quadratic eigenvalue problems. M.B. Van Gijzen's work in [12] sheds light on the limitations of linearization and proposes an alternative approach, known as Jacobi-Davidson (JD) method, for solving QEPs derived from an acoustic problem with damping. It

is demonstrated that the Jacobi-Davidson algorithm is efficient in approximating the smallest eigenpair and also possesses good parallelization capabilities. In Chapter 3, we will discuss further details regarding the methodology for solving the quadratic eigenvalue problems.

For some specific potentials between two quantum particles, there are analytic solutions available for the two-body system. The Pöschl-Teller potential stands as an exact example of this scenario. Cevik et al. (2016) [9] analyze the one-dimensional scattering produced by all variations of the Pöschl-Teller potential (see Eq. (1) in [9]), i.e., potential well, low, and high barriers. The analytic solutions for the bound, antibound, and resonant states of the system governed by the Pöschl-Teller potential have been demonstrated. These solutions provide explicit formulas for the eigenvalues and corresponding wavefunctions of the Schrödinger equation, which can be compared against our numerical results. In addition to [9], Moiseyev et al. (1998) [20] also present the analytic solutions to a specific potential, the universal energy-independent complex absorbing potential (CAP). Table 1 presented in [20] shows the bound state and resonance positions and widths obtained by using the universal energy-independent flux-diffusion type CAP (see Eq. (8) in [20]). The analytic solutions can serve as valuable benchmarks with which we can compare the numerical methods. In Subsection 5.1.1, we will compare our numerical results with the analytic solutions of the Pöschl-Teller potential to validate the correctness of our numerical algorithm.

1.3. Research questions

Having gained a foundational knowledge of computing resonant states of quantum few-body problems, several research questions remain ambiguous and require further investigation for clarification. These questions include:

1. Discretization of three-body equations

The whole process of spectral discretization of a two-body system has been shown by [29]. In Chapter 2, we will delve deeper into the topic of discretization for the two-body system. However, the discretization process for a three-body problem is not as straightforward as in the two-body case due to the introduction of a new Jacobi coordinate y between two heavy particles. In the absence of specified boundary conditions, [27] simply makes use of the tensor product formulation to derive the three-body Hamiltonian. However, the outgoing boundary conditions of resonances make performing the tensor product more tricky. We need to answer if the discretization has to be adapted to preserve the tensor structure of the Hamiltonian operator in the presence of boundary conditions. Can the Kronecker product be simply applied to the Hamiltonian system with boundary conditions?

2. Comparison of eigensolvers

The discretization of few-body resonant systems results in a quadratic eigenvalue problem. Two prominent options for solving QEPs can be summarized through the literature review: linearization and the Jacobi-Davidson method. The Jacobi-Davidson method is capable of directly solving the QEP, while the linearization approach requires collaboration with another generalized eigensolver. Analogous to the work presented in [27], where JDQR is employed to solve standard eigenvalue problems, it is also possible to apply iterative solvers to address the generalized eigenvalue equations resulting from the linearization process, for which we choose the Jacobi-Davidson QZ (JDQZ) algorithm. Therefore, we will solve the QEPs by means of two approaches respectively: the linearization method coupled with the Jacobi-Davidson QZ algorithm, and the (quadratic) Jacobi-Davidson algorithm. To determine a better solver for some specific problem sizes, our question is: Which one can converge faster? Which one has better scalability?

3. Preventing Repetition of Solutions

Later in Chapter 3, we will introduce another significant challenge faced by the Jacobi-Davidson method, which is the search for multiple distinct eigenvalues. The Jacobi-Davidson method has the chance to converge repeatedly to the same eigenvalue. This repetitive behavior raises the

question: How can we ensure that each convergence of the Jacobi-Davidson algorithm leads to a different eigenvalue?

4. Preconditioning

The iterative solvers from the Jacobi-Davidson family, such as JD, JDQR, and JDQZ, can achieve improved convergence speed through the application of preconditioning. In Section 3.2, we will demonstrate that the most computationally intensive part of the Jacobi-Davidson algorithm is solving the correction equation in every iteration. To achieve a decent performance of the Jacobi-Davidson algorithm, preconditioning of the correction equation becomes necessary. The selection of an appropriate preconditioner depends on the specific problems that need to be solved. [27] proposes a shifted Hamiltonian without potential as a good preconditioner for JDQR when computing few-body bound states. Can a preconditioner analogous to the one proposed for the bound state computation in [27] be derived for the resonant state computation?

5. Parallelization

We will test our eigensolvers on quantum two-body systems. However, in the case of three-body problems, the increase in dimension leads to a significantly larger number of grid points. The enormous size of three-body problems necessitates extensive computational power to handle the computational demands effectively. Therefore, another crucial task is the implementation of the eigensolver on supercomputers, specifically the parallelization of the Jacobi-Davidson algorithm. Our objective is to develop an optimal strategy for implementing the parallel JD algorithm to leverage the computational power of supercomputers. After implementing the parallel algorithm, we need to analyze its performance, such as speedup.

1.4. Thesis outline

This thesis is structured into several chapters to provide a comprehensive exploration of the research topic. The contents of the following chapters are as follows:

2. Discretization
   This chapter delves into the process commonly referred to as discretization, which involves converting the equations derived from quantum physics into algebraic equations that can be solved numerically. We will discuss and compare two approaches based on pseudo-spectral discretization, namely the Galerkin method and the Chebyshev differentiation matrix, and choose one of them to discretize the problem based on their performance in computational efficiency and ease of implementation. In the final section, we will present the entire process of discretizing both the two-body and one-dimensional three-body problems.

3. Quadratic Eigenvalue Problems
   Chapter 3 provides the methodology employed to solve the quadratic eigenvalue problems that arise from discretizing few-body problems. It covers a conventional approach known as linearization and an iterative eigensolver called the Jacobi-Davidson method. We will introduce various extensions of the Jacobi-Davidson algorithm that can address standard, generalized, and quadratic eigenvalue problems, respectively. We will delve into the underlying mathematics behind these extensions, shedding light on their implementation.

4. Parallelization
   In Chapter 4, we discuss the implementation details of hybrid distributed/shared memory parallelization of the Jacobi-Davidson algorithm on supercomputers. The methodology for parallelization, along with accompanying source code templates, will be provided. Based on the tensor structure outlined in Chapter 2, we present an efficient tensor product scheme that facilitates the high-performance application of stiffness and damping operators in three-body problems.

5. Numerical Results
   In Chapter 5, we showcase the numerical results obtained from our eigensolvers. In the scenario

of solving a two-body test problem, we compare the performance of the JD and JDQZ method in terms of convergence speed and computational time. After analyzing the performance of the Jacobi-Davidson method in solving two-body problems, we proceed to present its performance in 1D three-body problems and the resonance poles obtained. Research questions are addressed within the context of computing resonant states of both two-body and three-body problems in this chapter.

6. Discussion and Conclusion
   The final chapter of the thesis serves as a conclusion and evaluation of the research. We conclude the answers to the research questions and address the limitations of our current research. Furthermore, we briefly introduce the potential applications of our study and explore potential avenues for future improvements. The conclusion section summarizes the key contributions of the study, highlighting the main accomplishments and advancements made in the field of numerical solvers for quantum mechanical few-body problems.

2

Discretization

The discretization techniques enable us to transform the continuous problem into an algebraic form that can be solved numerically. In this chapter, two discretization approaches derived from the pseudo-spectral method are introduced and discussed. From these two methods, we will carefully select and utilize one approach to discretize and formulate our two-body and three-body problems.

2.1. Pseudo-spectral methods

To reduce the equations to a discrete algebraic form, numerous numerical methods are available, such as the finite difference method, the finite element method, the spectral method, and so on. Throughout the research, we follow the discretizing procedure in [14] and [27], using a variant of the spectral methods, the pseudo-spectral methods [8, 24, 30].

2.1.1. Comparison with FEM/FDM

The pseudo-spectral methods are closely related to the spectral methods. In fact, they can be considered spectral methods with a pseudo-spectral basis, which allows the representation of functions on a quadrature grid. The idea of the spectral methods is to denote the solution of the differential equation as a sum of certain basis functions and then to choose the coefficients in the sum in order to satisfy the differential equation as well as possible. There is a wide range of choices for basis functions, such as sinusoids or polynomial functions.

Spectral algorithms are in philosophy similar to finite element methods (FEM). The major difference is that finite elements chop the interval in x into several sub-intervals, and choose $\pi_n(x)$ to be local functions which are polynomials of fixed degree which are non-zero only over a couple of sub-intervals. In contrast, spectral methods use global basis functions in which $\pi_n(x)$ is a polynomial (or trigonometric polynomial) of high degree which is non-zero, except at isolated points, over the entire computational domain. Finite elements give rise to sparse systems which can be solved at a fraction of the cost of problems of “full” matrices with similar size. In addition, in multi-dimensional problems, the little sub-intervals become little triangles that can be fitted to irregularly-shaped bodies. However, the disadvantage of the finite element method is also significant: the accuracy is low because each basis function is a polynomial of a low degree. In contrast, spectral methods generate algebraic equations with dense matrices, but in compensation, the high order of the basis functions gives high accuracy for a given N. When fast matrix-solvers are used, spectral methods can be much more efficient than finite element or finite difference methods for many classes of problems such as wave propagation problems [19].

Compared with the finite difference method (FDM), spectral methods can perform more accurately and save more memory [8]. Finite difference methods utilize a series of overlapping polynomials to approximate the unknown function $f(x)$ by interpolating it at various grid points. For a three-point

central difference, the approximation to the derivative reads

$$\frac{\mathrm{d}f}{\mathrm{d}x} \approx \frac{f(x+h) - f(x-h)}{2h} + O(h^2). \tag{2.1}$$

Since the interval h is $O(1/N)$, where N is the number of grid points, we can conclude that the error is $O[(1/N)^2]$. [8] analyzes that in pseudo-spectral methods, the order of error is not fixed but rather varies with N, given by

$$\textit{Pseudo-spectral error} \approx O[(1/N)^N]. \tag{2.2}$$

The error (2.2) is decreasing faster than any order of finite difference. To verify the argument, we employ both pseudo-spectral methods and finite difference methods to solve a 1D Poisson's equation with Dirichlet boundary conditions

$$\frac{\mathrm{d}^2u}{\mathrm{d}x^2} = e^{4x},\ x \in (-1, 1), \tag{2.3}$$

$$u(-1) = u(1) = 0, \tag{2.4}$$

the analytic solution of which is given by $(e^{4x} - (\sinh 4)x - \cosh 4)/16$. Figure 2.1 depicts the l_2 error for both methods. An exponential convergence of the pseudo-spectral error can be observed, which is significantly faster compared to the finite difference method.

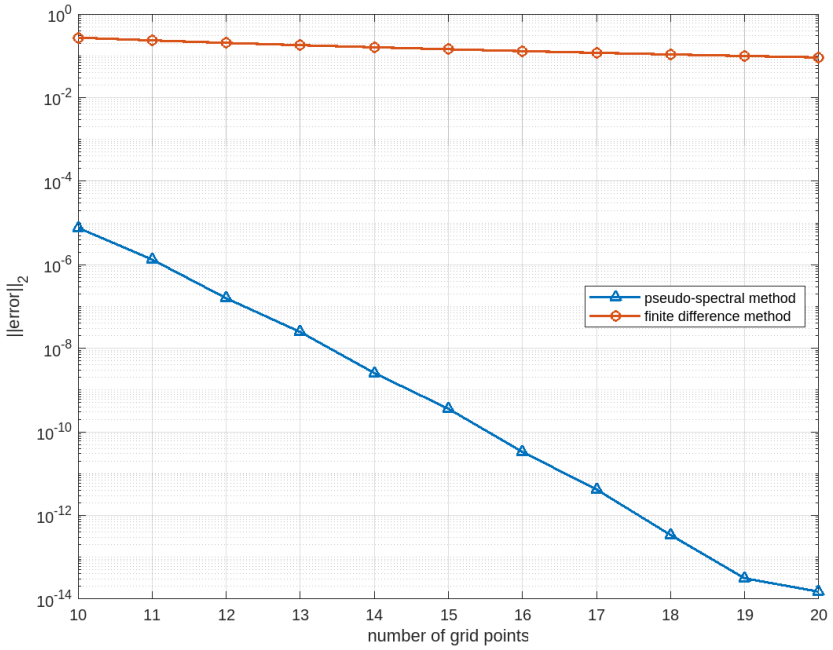


Figure 2.1: l_2 error of the pseudo-spectral and finite difference methods as a function of the number of grid points.

Following the discussion of various discretization methods, we are now embarking on our exploration of how to utilize pseudo-spectral methods for discretizing our problem. We again consider a one-dimensional wave function in a two-body system, expanded by basis as

$$\psi(x) = \sum_{n=1}^{N} c_n \pi_n(x),\ -L \leq x \leq L, \tag{2.5}$$

where $\pi_n(x)$ are basis functions and c_n are coefficients for every $\pi_n(x)$. By substituting $\psi(x)$ with (2.5) for Eq. (1.14), we can derive a stationary Schrödinger equation represented by basis functions with coefficients. There are two directions to discretization: integration and differentiation. The boundary conditions can be automatically incorporated into the equation by integrating the Schrödinger equation over the domain, known as the Galerkin method. Another approach involves utilizing the Chebyshev differentiation matrix, derived from the pseudo-spectral collocation method, to discretely represent the derivative. In the subsequent subsections, we will discuss these two approaches and choose one of them to discretize our problem.

2.1.2. Galerkin method

In the present subsection we introduce a general derivation proposed by [29] which deduces a finite basis representation from the continuous Schrödinger equation by integrating the equation over the domain. We again apply a one-dimensional finite basis

$$\pi_m(x),\ i = 1, \cdots, N. \tag{2.6}$$

By multiplying Eq. (1.14) by $\pi_m(x)$ and integrating over $x \in [-L, L]$, under the boundary conditions (1.15) we can derive

$$\int_{-L}^{L} \pi_m(x)[-\frac{1}{2}\frac{\mathsf{d}^2}{\mathsf{d}x^2} + V(x) - \frac{1}{2}k^2]\psi(x)\mathsf{d}x = 0, \tag{2.7}$$

which is equivalent to

$$\begin{aligned} &-\frac{ik}{2}[\pi_m(L)\psi(L) + \pi_m(-L)\psi(-L)] + \frac{1}{2}\int_{-L}^{L} \frac{\mathsf{d}\pi_m(x)}{\mathsf{d}x}\frac{\mathsf{d}\psi(x)}{\mathsf{d}x}\mathsf{d}x + \\ &\int_{-L}^{L} \pi_m(x)[V(x) - \frac{1}{2}k^2]\psi(x)\mathsf{d}x = 0. \end{aligned} \tag{2.8}$$

By expanding $\psi(x)$ in terms of the basis (2.6) and substituting the expansion into Eq. (2.8), the original equation (2.8) is reduced to the matrix-represented quadratic eigenvalue equation

$$(K + \frac{ik}{2}C - \frac{k^2}{2}M)\mathbf{c} = 0, \tag{2.9}$$

where entries of K, C, and M are given by

$$K_{mn} = \frac{1}{2}\int_{-L}^{L} \frac{\mathsf{d}\pi_m(x)}{\mathsf{d}x}\frac{\mathsf{d}\pi_n(x)}{\mathsf{d}x}\mathsf{d}x + \int_{-L}^{L} \pi_m(x)V(x)\pi_n(x)\mathsf{d}x, \tag{2.10}$$

$$C_{mn} = \pi_m(L)\pi_n(L) + \pi_m(-L)\pi_n(-L), \tag{2.11}$$

$$M_{mn} = \int_{-L}^{L} \pi_m(x)\pi_n(x)\mathsf{d}x, \tag{2.12}$$

and **c** is a vector whose entries are coefficients of the basis functions $\pi_n(x)$. If the basis $\pi_n(x)$ is orthonormal on the domain $[-L, L]$, M is always the identity matrix. The quadratic eigenvalue equation (2.9) is a projection of Eqs. (1.14) and (1.15) onto an N-dimensional Hilbert space spanned by the basis $\pi_m(x)$.

The derivation above is exactly the idea of the Galerkin method. Given the formula of every entry of each matrix, we can compute K, C, M respectively and finally derive a matrix representation of the quadratic eigenvalue equation. Although the Galerkin method is quite powerful, the integrals required to construct the matrix elements can be prohibitively complicated, especially when some complex basis functions are applied. We move forward to the pseudo-spectral methods to save us from calculating such complex integrals analytically. The idea is that if $\{\pi_n\}$ is a polynomial basis, such as the Chebyshev polynomial, then the inner products required by the Galerkin method can be accurately approximated by Gaussian quadrature.

Definition 4. *The Chebyshev polynomial of the first kind is given by*

$$T_n(x) = \mathsf{cos}(n\theta),\ x = \mathsf{cos}(\theta), \tag{2.13}$$

which is obtained from the recurrence relation

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_n(x) &= 2xT_{n-1} - T_{n-2}(x). \end{aligned}$$

Consider the set $\{T_n(x)\}$ of orthogonal Chebyshev polynomials of the first kind on the interval $[-1, 1]$. Some lower-order Chebyshev polynomials are given by

$$T_0(x) = 1,\ T_1(x) = x,\ T_2(x) = 2x^2 - 1,\ T_3(x) = 4x^3 - 3x,$$
$$T_4(x) = 8x^4 - 8x^2 + 1.\ T_5(x) = 16x^5 - 20x^3 + 5x.$$

With $x = \cos(\theta)$, the Chebyshev polynomials are orthogonal in accordance with

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} T_m(x) T_n(x) \mathrm{d}x = c_n \frac{\pi}{2} \delta_{mn} \tag{2.14}$$

where

$$c_n = \begin{cases} 2, & n = 0, \\ 1, & n \geq 1. \end{cases}$$

The quadrature points of order N for the basis set are determined as the roots of $\cos(N\theta_i) = 0$ given by $\theta_i = (2i-1)\pi/2N,\ i = 1, \cdots, N$.

Definition 5. *The Gauss-Chebyshev quadrature points are given by*

$$x_i = \cos[(2i-1)\frac{\pi}{2N}],\ i = 1, 2, \cdots, N. \tag{2.15}$$

Noting the normalization of the Chebyshev polynomials, the quadrature weights are given by

$$w_i = -\frac{\pi}{T_{N+1}(x_i) T'_N(x_i)}, \tag{2.16}$$

where

$$T'_N(x) = N \sin(N\theta)/\sin(\theta),$$
$$T_{N+1}(x_i) = \cos[(N+1)\theta_i] = -\sin(\theta_i).$$

With $\cos(N\theta_i) = 0$ and $\sin(N\theta_i) = 1$, we have $T_{N+1}(x_i) T'_N(x_i) = -N$. Thus we can rewrite the weights in the quadrature (2.16) to

$$w_i = \frac{\pi}{N}. \tag{2.17}$$

With Eqs. (2.15), (2.16), we can derive the Gauss-Chebyshev quadrature.

Definition 6. *The Gauss-Chebyshev quadrature is a Gaussian quadrature over the interval* $[-1, 1]$ *with weighting function* $w(x) = (1-x^2)^{-1/2}$*:*

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} f(x) \mathrm{d}x \approx \sum_{i=1}^{N} w_i f(x_i). \tag{2.18}$$

Note that so far the integral is limited in the interval $[-1, 1]$. To generalize Eq. (2.18) to an arbitrary interval such as $[-L, L]$, we need to apply linear transformation to x:

$$\tilde{x} = Lx = L\cos(\theta),\ \tilde{x} \in [-L, L].$$

Then we derive the Gauss-Chebyshev quadrature in the interval $[-L, L]$.

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} f(x) \mathrm{d}x = \int_{-L}^{L} \frac{1}{\sqrt{L^2 - \tilde{x}^2}} f(\frac{\tilde{x}}{L}) \mathrm{d}\tilde{x} \approx \sum_{i=1}^{N} \frac{\pi}{N} f(x_i) \tag{2.19}$$

With Eq. (2.19), we can now approximate the integrals in the formulas for K and M in Eq. (2.10) and (2.12) using the following formulas:

$$K_{mn} = \sum_{i=1}^{N} \frac{\pi}{N} \sqrt{L^2 - x_i^2} [\frac{1}{2} T'_m(Lx_i) T'_n(Lx_i) + T_m(Lx_i) V(Lx_i) T_n(Lx_i)], \tag{2.20}$$

$$M_{mn} = \sum_{i=1}^{N} \frac{\pi}{N} \sqrt{L^2 - x_i^2} T_m(Lx_i) T_n(Lx_i), \tag{2.21}$$

where $x_i = \cos[(2i-1)\pi/2N],\ i = 1, \cdots, N$. By means of the pseudo-spectral methods, we convert the original integral (2.10), (2.12) into the sum of the functions evaluated at Gauss-Chebyshev quadrature points. Nevertheless, computing such summation for every entry still requires much effort. In order to avoid the computation of the summation, we can resort to an alternative direction of discretization, the Chebyshev differentiation matrix.

2.1.3. Chebyshev differentiation matrix

The Chebyshev differentiation matrix [8, 30] that results from the high-order pseudo-spectral collocation method is a matrix operator representing the derivative. We introduce the Gauss-Chebyshev-Lobatto points (Chebyshev points for short), which are different from the Gauss-Chebyshev quadrature points in the previous subsection, in $x \in [-1, 1]$.

Definition 7. *The Gauss-Chebyshev-Lobatto points are given by*

$$x_j = \cos(j\pi/N),\ j = 0, 1, \cdots, N. \tag{2.22}$$

In this subsection, we shall use these points to construct Chebyshev differentiation matrices and apply these matrices to differentiate the wave function ψ. Given a grid function u defined on the Chebyshev points, we obtain a discrete derivative w in two steps:

- *Let p be the unique polynomial of degree $\leq N$ with $p(x_j) = \psi_j$, $0 \leq j \leq N$.*
- *Set $w_j = p'(x_j)$.*

The operation is linear. Hence it can be represented by multiplication by an $(N+1) \times (N+1)$ matrix, which we can denote by D_N:

$$w_i = (D_N)_{ij}\psi_j, \tag{2.23}$$

where N is an arbitrary positive integer and $(D_N)_{ij}$ represents the (i, j) elements of the matrix D_N. In order to derive the differentiation matrix D_N, one needs to consider the interpolation polynomial and the interpolation coefficients. A detailed derivation will not be given here as it's not the focus of the research. We simply show the result of the first-order Chebyshev differentiation matrix given by [30]:

Theorem 1. *For the grid (2.22) and each $N > 1$, let the rows and columns of the $(N+1) \times (N+1)$ Chebyshev spectral differentiation D_N be indexed from 0 to N. The entries of this matrix are:*

$$\begin{aligned}
(D_N)_{00} &= \frac{2N^2+1}{6}, \qquad (D_N)_{NN} = -\frac{2N^2+1}{6}, \\
(D_N)_{jj} &= \frac{-x_j}{2(1-x_j^2)}, \quad j = 1, \cdots, N-1, \\
(D_N)_{ij} &= \frac{\gamma_i}{\gamma_j}\frac{(-1)^{i+j}}{(x_i - x_j)}, \quad i \neq j, \quad i, j = 1, \cdots, N-1,
\end{aligned}$$

where

$$\gamma_i = \begin{cases} 2, & i = 0 \text{ or } N \\ 1, & otherwise. \end{cases}$$

The whole pattern of the matrix is shown in figure 2.2. So far we have already defined the Chebyshev differentiation matrix D_N which can represent the first-order derivative from the matrix side. But recall that our problem contains not only the first-order derivative but also the second-order derivative and the boundary conditions. To construct a matrix operator of $\mathrm{d}^2/\mathrm{d}x^2$, we can compute the square of D_N, D_N^2. D_N^2 can be evaluated either by squaring D_N, which costs $O(N^3)$ floating point operations or by explicit formula or recurrences. Applying explicit formulas or recurrences is computationally cheaper, requiring $O(N^2)$ flops. Whether to use square or explicit formula depends on the problem size.

In practice, we usually do not compute D_N exactly using the explicit formulas given by Theorem 1. A small trick in the implementation of D_N is to utilize explicit formulas for the off-diagonal entries but

then obtain the diagonal entries from

$$(D_N)_{ii} = -\sum_{j=0, j\neq i}^{N} (D_N)_{ij}. \tag{2.24}$$

In [3], Baltensperger and Berrut propose that the application of Eq. 2.24 can provide a higher precision by reducing the roundoff errors in comparison to the standard formula.

$$D_N = \begin{array}{|c|c|c|}
\hline
\frac{2N^2+1}{6} & 2\frac{(-1)^j}{1-x_j} & \frac{1}{2}(-1)^N \\
\hline
-\frac{1}{2}\frac{(-1)^i}{1-x_i} & \frac{(-1)^{i+j}}{x_i-x_j} \quad \frac{-x_j}{2(1-x_j^2)} \quad \frac{(-1)^{i+j}}{x_i-x_j} & \frac{1}{2}\frac{(-1)^{N+i}}{1+x_i} \\
\hline
-\frac{1}{2}(-1)^N & -2\frac{(-1)^{N+j}}{1+x_j} & -\frac{2N^2+1}{6} \\
\hline
\end{array}$$

Figure 2.2: The whole pattern of the Chebyshev spectral differentiation matrix D_N.

The utilization of the Chebyshev differentiation matrix is fairly easy. Assuming a grid comprising $N+1$ Chebyshev points denoted as $x_0, x_1, \ldots, x_N$ on the x-axis, we can represent the first-order and second-order derivatives of the wave function in Eq. (1.1) as follows:

$$\frac{\mathrm{d}}{\mathrm{d}x}\psi(x) \to D_N\boldsymbol{\psi},$$
$$\Delta_x\psi(x) \to D_N^2\boldsymbol{\psi},$$

where $\boldsymbol{\psi} = \{\psi_0, \psi_1, \cdots, \psi_N\}^T$ corresponds to the wave function $\psi(x)$ evaluated at the grid points $x_0, x_1, \cdots, x_N$. Compared with the Galerkin method, applying the Chebyshev differentiation matrix is much easier from the programming side and also requires less computation. Our research chooses to utilize the Chebyshev differentiation matrix to formulate the two-body and three-body problems, of which more details about integrating two derivatives into one equation will be presented in the next section.

2.2. Application to few-body problems

With the discretization method at our disposal, we are now capable of formulating the problem that will be solved in our research. Subsection 2.2.1 presents a comprehensive pseudo-spectral discretization procedure for formulating two-body test problems. Subsequently, in Subsection 2.2.2, we aim to answer the first research question, extending the discretization methodology to three-body problems through Kronecker products.

2.2.1. Two-body problems

With the Chebyshev differentiation matrix, we have only one step remaining to formulate the problem: integrating the Schrödinger equation and boundary conditions into a single equation. For the boundary

condition (1.15), we can proceed as follows: Consider a grid of Chebyshev points $x_0, \cdots, x_N$ and the corresponding solution to the wave function $\psi_0, \cdots, \psi_N$. We take the interior Chebyshev points $x_1, \cdots, x_{N-1}$ as our computational grid computing the equation (1.14):

$$\begin{aligned} &[-\frac{1}{2}\frac{\mathsf{d}^2}{\mathsf{d}x^2} + V(x) - \frac{1}{2}k^2]\psi(x) = 0 \\ &\qquad\qquad \downarrow \\ &[-\frac{1}{2}D_N^2 + V - \frac{1}{2}k^2 I]\boldsymbol{\psi} = 0, \end{aligned} \tag{2.25}$$

where V is the diagonal matrix whose entries are values of the potential $V(x)$ evaluated at $x_1, \cdots, x_{N-1}$ and I is the identity matrix. $\boldsymbol{\psi}$ is the vector that indicates the values of the wave function $\psi(x)$ at grid points $x_1, \cdots, x_{N-1}$. Note that the order of Chebyshev grid points is reversed with respect to the real axis [-1,1], i.e., [1,-1] in actual. And hence at two end points x_0 and x_N, we impose the boundary condition (1.15) by means of the first-order Chebyshev differentiation matrix:

$$\begin{aligned} &(D_N)_{0,:}\psi = ik\psi_0, \\ &(D_N)_{N,:}\psi = -ik\psi_N, \end{aligned} \tag{2.26}$$

where $(D_N)_{0,:}$ and $(D_N)_{N,:}$ denote the first row and the last row of the differentiation matrix respectively. Note that the derivation above is based on the domain $[-1, 1]$ and hence if we want to generalize the problem to $x \in [-L, L]$, a linear transformation to the coordinate is needed. We can easily extend Eq. (2.27) to an arbitrary cutoff length L through a linear transformation:

$$\begin{aligned} &\text{Chebyshev grid x} \to Lx, \\ &\text{Differentiation matrix } D_N \to D_N/L. \end{aligned}$$

By combining Eq. (2.25) and Eq. (2.26) together, we can include the boundary condition inside the quadratic eigenvalue equation. The QEP for a two-body system is formulated as:

$$\begin{aligned} &K^{(2b)}\boldsymbol{\psi} + kC^{(2b)}\boldsymbol{\psi} + k^2M^{(2b)}\boldsymbol{\psi} = 0, \\ &where \quad (K^{(2b)})_{0,:} = (D_N)_{0,:}, \quad (K^{(2b)})_{N,:} = -(D_N)_{N,:}, \quad (K^{(2b)})_{1:N-1,:} = (\frac{1}{2}D_N^2 - V)_{1:N-1,:}, \\ &C^{(2b)} \textit{ is 0 everywhere except } (C^{(2b)})_{0,0} = (C^{(2b)})_{N,N} = -i, \\ &M^{(2b)} \textit{ is a diagonal matrix with diagonal entries } 0, 1/2, 1/2, \cdots, 1/2, 1/2, 0. \end{aligned} \tag{2.27}$$

Based on the equation (2.27), it can be observed that the matrix polynomial $K^{(2b)} + kC^{(2b)} + k^2M^{(2b)}$ is nonsingular, while both the mass matrix M and the damping matrix C are singular. The mass matrix M is symmetric positive semi-definite. In the case where M is nonsingular, a quadratic eigenvalue problem of size N yields $2N$ finite eigenvalues. However, if M is singular, both finite and infinite eigenvalues are present. In the upcoming chapter, we will demonstrate that the singularity of matrix M introduces certain constraints to the linearization procedure, which is an approach we will introduce for solving QEPs. In Chapter 5, we will demonstrate how we utilize the two-body quadratic eigenvalue problem given by equation (2.27) as a test problem to evaluate the performance of our eigensolvers.

2.2.2. Three-body problems

The discretization of the three-body equation (1.16) and the boundary conditions (1.17) corresponds to our first research question. To address the multi-dimensional system, [27] applies the Kronecker product to discretize the Schrödinger equation without boundary conditions. This approach is commonly used to discretize higher-dimensional problems. Taking inspiration from the scenario of bound states in [27], a natural idea is to utilize the Kronecker product for discretizing the three-body resonance as well. According to our first research question, we need to answer if the discretization has to be adapted to preserve the tensor structure of the Hamiltonian operator in the presence of boundary conditions.

Definition 8. *If A is an $m \times n$ matrix and B is a $p \times q$ matrix, then the Kronecker product $A \otimes B$ is the*

$pm \times qn$ ***block matrix:***

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}. \tag{2.28}$$

The Kronecker product represents a specialized form of the tensor product that extends from vectors to matrices and provides the matrix of the tensor product linear transformation based on a standard basis selection. It has the following useful properties:

(a) $(A \otimes B)(C \otimes D) = (AC \otimes BD)$ for matrices $A,\ B,\ C,\ D$ of conforming dimensions,
(b) $(A \otimes B)^T = A^T \otimes B^T$,
(c) $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$, if both A and B are invertible.

The answer to the first research question is affirmative. It is worth noting that the application of Kronecker products directly to two-body operators is not appropriate when boundary conditions are present. Before proceeding with the discretization process, it is essential to know the specific equations to which the wave function conforms within different regions of the domain. Figure 2.3 illustrates three distinct spatial relationships that can arise between three particles within a square 2D domain $[-L, L]^2$, each exhibiting unique behaviors.

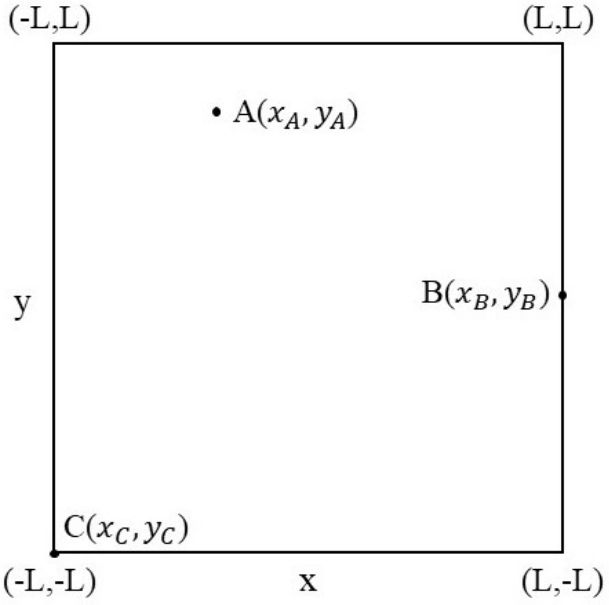


Figure 2.3: Three possible configurations of spatial coordinates (x, y).

A. Point A, located inside the domain $[-L, L]^2$, adheres to the stationary Schrödinger equation stated by Eq. 1.5

$$[-\frac{\alpha_x}{2}\Delta_x - \frac{\alpha_y}{2}\Delta_y + V(x_A, y_A)]\psi(x_A, y_A) = \frac{1}{2}k^2\psi(x_A, y_A). \tag{2.29}$$

B. Point B lies on the boundary line of the domain, and it also adheres to the stationary Schrödinger equation 1.5. However, it is crucial to consider the outgoing boundary condition on the x-coordinate as well. Therefore, the equations used to describe B are given by

$$[-\frac{\alpha_x}{2}\Delta_x - \frac{\alpha_y}{2}\Delta_y + V(x_B, y_B)]\psi(x_B, y_B) = \frac{1}{2}k^2\psi(x_B, y_B), \tag{2.30}$$

$$(\frac{\partial}{\partial x} - ik)\psi(x_B, y_B) = 0. \tag{2.31}$$

Through differentiation Eq. (2.31) can be reduced to

$$\Delta_x \psi(x_B, y_B) = ik \frac{\partial}{\partial x} \psi(x_B, y_B). \tag{2.32}$$

By substituting Eq. (2.32) into Eq. (2.30), we can derive a unified equation

$$[-\frac{\alpha_x}{2}(ik\frac{\partial}{\partial x}) - \frac{\alpha_y}{2}\Delta_y + V(x_A, y_A)]\psi(x_A, y_A) = \frac{1}{2}k^2\psi(x_A, y_A) \tag{2.33}$$

that satisfies both the Schrödinger equation (2.30) and the boundary condition (2.31).

C. Point C represents a corner point where both coordinates lie on the boundary of the domain. Hence the point should satisfy the boundary conditions on both coordinates. The unified equation can be obtained analogously to point B, yielding

$$[\frac{\alpha_x}{2}(ik\frac{\partial}{\partial x})\frac{\alpha_y}{2}(ik\frac{\partial}{\partial y}) + V(x_C, y_C)]\psi(x_C, y_C) = \frac{1}{2}k^2\psi(x_C, y_C). \tag{2.34}$$

In the quadratic eigenvalue equations (2.29), (2.33), and (2.34), the coefficient of the quadratic term remains unchanged. As a result, the mass matrix M in the three-body discretization remains a diagonal matrix, with each entry equal to $1/2$. Before applying the Kronecker product, let us first express the stiffness matrix and damping matrix along the x dimension exclusively. The stiffness matrix K_x represents the second-order partial derivative of the wave function evaluated at non-boundary grid points on the x-axis. It is given by

$$(K_x)_{0,:} = \mathbf{0}^T, \;\; (K_x)_{N,:} = \mathbf{0}^T, \;\; (K_x)_{1,N-1:} = -\frac{\alpha_x}{2}(D^2_{N_x})_{1:N-1,:}, \tag{2.35}$$

where $\mathbf{0}^T$ denotes the transpose of the zero vector and D_{N_x} is the Chebyshev differentiation matrix after linear transformation. The damping matrix C_x, standing for the coefficient of the first-degree term, is obtained as follows

$$(C_x)_{0,:} = -i\frac{\alpha_x}{2}(D_{N_x})_{0,:}, \;\; (C_x)_{N,:} = i\frac{\alpha_x}{2}(D_{N_x})_{N,:}, \;\; (C_x)_{1:N-1,:} = O, \tag{2.36}$$

where O denotes a zero matrix. The operators K_y and C_y along the y-coordinate can be derived using the same procedure as in Eq. (2.35) and (2.36). By applying the Kronecker product to $K_{x,y}$ and $C_{x,y}$, the quadratic eigenvalue equations of three-body discretization reads

$$\begin{aligned}
&K^{(3b)}\boldsymbol{\psi} + kC^{(3b)}\boldsymbol{\psi} + k^2M^{(3b)}\boldsymbol{\psi} = 0, \textit{ where}\\
&K^{(3b)} = K_x \otimes \mathbb{I}_y + \mathbb{I}_x \otimes K_y + V,\\
&C^{(3b)} = C_x \otimes \mathbb{I}_y + \mathbb{I}_x \otimes C_y,\\
&M^{(3b)} \textit{ is a diagonal matrix with diagonal entries equal to } -1/2.
\end{aligned} \tag{2.37}$$

Eq. (2.37) formulates the QEP of 1D three-body problems to be solved by our research. The eigenvector

$$\boldsymbol{\psi} = \{\psi_{0,0}, \psi_{0,1}, \cdots, \psi_{0,N_y}, \psi_{1,0}, \cdots, \psi_{N_x,N_y}\}^T. \tag{2.38}$$

corresponds to the wave function $\psi(\vec{x}, \vec{y})$ in Eq. (1.5) evaluated at the grid points $(x^{(i)}, y^{(j)})$, yielding the entries $\psi_{i,j} = \psi(x^{(i)}, y^{(j)})$ with $i = 0, 1, \cdots, N_x$ and $j = 0, 1, \cdots, N_y$. Here $V = v_0(F_+ + F_-)$ is a diagonal matrix resulting from evaluating the potential function $f(|\vec{x} \pm \vec{y}/2|)$ at the corresponding grid points. It is worth noting that on the domain boundary, no potential term should be considered. But for simplicity, the situation of the potential on the boundary points is not taken into consideration when evaluating $f(|\vec{x} \pm \vec{y}/2|)$ in this context. This is because the potential function approximates zero on the boundary points when the cutoff distance is sufficiently large.

Some remarks on the discretization of the three-body problem: The discretization we employ in this subsection actually leads to an overdetermined problem. Indeed, the idea behind Eq. (2.33) and (2.34) is that the three-body resonant system simultaneously satisfies both the Schrödinger equation and the

Siegert boundary conditions on boundaries. The multiple constraints give rise to an overdetermined problem. In this case, the mass matrix M is nonsingular, indicating that all eigenvalues of the QEP are finite. The application of the Kronecker product transforms the dense problem of a two-body system into a sparse problem, leading to a significant expansion in the problem size. This transformation enables us to solve the higher dimensional problem, thereby increasing the computational complexity and memory requirements for solving the problem. But things are not that bad. For two dense matrices $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{p\times q}$, if $T = A \otimes B$ has a tensor structure, we only need to save A and B. This only occupies

$$\text{constant} \cdot (nm + pq) \text{ bytes},$$

while storing the full matrix needs constant$\cdot(nmpq)$ bytes. In Chapter 4, we will further demonstrate how to exploit the tensor structure of the operators to significantly reduce the computational complexity.

3

Quadratic Eigenvalue Problems

Our focus now turns to exploring strategies for solving the quadratic eigenvalue problems proposed previously. A key goal of the research is to develop an efficient solver to solve the QEPs. We will start by introducing a classical approach called linearization for solving QEPs, which transforms the quadratic problem into a generalized eigenvalue problem. Following the linear transformation, we apply the Jacobi-Davidson method, an iterative eigensolver, to solve the linearized problem. As an alternative, we could also use the Jacobi-Davidson method to approximate eigenpairs of the quadratic eigenvalue problem directly. In this chapter, we provide an outline of the Jacobi-Davidson process for solving both linear and quadratic problems.

3.1. Linearization

A quadratic eigenvalue problem is equivalent to a generalized eigenvalue problem or pencil, (A, B). The equivalence transformation is called linearization [12, 32], which is a conventional approach to compute quadratic eigenvalue problems. By means of linearization, we can transform the quadratic eigenvalue problem to a linear eigenvalue problem. In this section we discuss more details about such transformation including the basic scheme and its advantages and drawbacks.

Let's consider a quadratic eigenvalue problem in the form of

$$\lambda^2 M\mathbf{p} + \lambda C\mathbf{p} + K\mathbf{p} = 0 \tag{3.1}$$

The quadratic eigenvalue problem (3.1) can be transformed to an equivalent generalized eigenvalue problem

$$A\mathbf{x} = \lambda B\mathbf{x}. \tag{3.2}$$

Such transformation is called linearization, the most common and direct way to solve a QEP. If a QEP has some special structure, such as symmetry, the linearization can also have a related structure [32]. A standard linearization technique is shown below.

By considering $[\lambda\mathbf{p},\ \mathbf{p}]^\top$ as the eigenvector instead of $\mathbf{p}$, Eq.(3.1) accompanied by the identity $\lambda\mathbf{p} = \lambda\mathbf{p}$ gives the system

$$\begin{bmatrix} -C & -K \\ I & O \end{bmatrix} \begin{bmatrix} \lambda\mathbf{p} \\ \mathbf{p} \end{bmatrix} = \lambda \begin{bmatrix} M & O \\ O & I \end{bmatrix} \begin{bmatrix} \lambda\mathbf{p} \\ \mathbf{p} \end{bmatrix}, \tag{3.3}$$

where I and O denote the identity matrix and zero matrix respectively. If M is invertible, Eq.(3.3) can be reduced to a standard eigenvalue problem

$$\begin{bmatrix} M^{-1} & O \\ O & I \end{bmatrix} \begin{bmatrix} -C & -K \\ I & O \end{bmatrix} \begin{bmatrix} \lambda\mathbf{p} \\ \mathbf{p} \end{bmatrix} = \lambda \begin{bmatrix} \lambda\mathbf{p} \\ \mathbf{p} \end{bmatrix}. \tag{3.4}$$

Eq.(3.3) has several different variants. One can multiply (3.3) by a matrix $\begin{bmatrix} I & O \\ O & -K \end{bmatrix}$ from left and derive a symmetric formulation

$$\begin{bmatrix} C & K \\ K & O \end{bmatrix} \begin{bmatrix} \lambda \mathbf{p} \\ \mathbf{p} \end{bmatrix} = \lambda \begin{bmatrix} -M & O \\ O & K \end{bmatrix} \begin{bmatrix} \lambda \mathbf{p} \\ \mathbf{p} \end{bmatrix}. \tag{3.5}$$

One can also move $-C$ in Eq.(3.3) to the right side,

$$\begin{bmatrix} O & -K \\ I & O \end{bmatrix} \begin{bmatrix} \lambda \mathbf{p} \\ \mathbf{p} \end{bmatrix} = \lambda \begin{bmatrix} M & C \\ O & I \end{bmatrix} \begin{bmatrix} \lambda \mathbf{p} \\ \mathbf{p} \end{bmatrix}. \tag{3.6}$$

The advantage of Eq.(3.5) is that if K and C are symmetric, then (3.5) is symmetric pencils. Regrettably, the formulations presented in (3.4), (3.5), and (3.6) do not apply to our problem. In Subsection 2.2.1, we have demonstrated the singularity of the mass matrix, rendering it non-invertible. Consequently, it is impossible to reduce our QEP to Eq. (3.4). Furthermore, the lack of symmetry in the stiffness matrix prohibits the utilization of symmetric pencils (3.5) and (3.6). As a consequence, in our problem, we are confined to utilizing the generalized linearization given by Eq. (3.3).

Implementing linearization such as Eq. (3.3) is easy, but has a significant drawback that doubles the matrix size. When dealing with problem sizes that are not large, one can simply employ the QZ decomposition method, such as utilizing the *eig* command in MATLAB, to effectively solve Eq. (3.3). However, in situations where the problem size is large, employing iterative eigensolvers becomes more efficient and preferable. Some Krylov-based iterative solvers, such as Arnoldi's or Lanczos's method with a shift-and-invert strategy, exhibit decent performance when computing the exterior eigenvalues of large systems, particularly for sparse systems. Besides the Arnoldi method (AR), there is another option for solving QEPs called the Jacobi-Davidson method. The Jacobi-Davidson method was originally designed for solving standard eigenvalue problems but could be extended for solving generalized and quadratic eigenvalue problems. [31] compared the running time of the JD method and the AR method on different problem sizes and concluded that JD is globally more efficient than AR, especially when the problem size is large and a small number of eigenpairs are sought for. Considering the size of our problem and the limited number of desired eigenpairs, we have chosen to use the Jacobi-Davidson QZ algorithm, a variant of the JD method, to approximate the eigenvalues of the linearized eigenvalue problem. Furthermore, besides JDQZ we will customize the JD method to solve the quadratic eigenvalue problem directly without any reduction. In the next section, we will provide all the necessary ingredients about the family of the Jacobi-Davidson method necessary for our research.

3.2. Jacobi-Davidson method

The Jacobi-Davidson method [2, 6, 26] is an iterative algorithm solving eigenvalue problems by iteratively approximating certain eigenvectors. The basic idea is to look for the best approximations to the eigenvectors in some search space. An excellent merit of the JD algorithm is that no inverting operations are involved and the computational cost of every single iteration is fairly small when the dimension of the search space is not big. This nature makes the JD method quite efficient in parallelism and solving large eigenvalue problems.

The Jacobi-Davidson algorithm can be divided into two parts, the Davidson algorithm and the Jacobi correction part, which will be introduced in the following Subsection 3.2.1 and 3.2.2 respectively. Although the JD method was initially proposed to solve standard eigenvalue problems, some extensions are addressing generalized and polynomial eigenvalue problems. JDQZ algorithm [11], one of the variants essential for solving generalized eigenvalue problems, is introduced in Subsection 3.2.3. Then in Subsection 3.2.4 we will show the possibility of using the JD algorithm to directly solve quadratic eigenvalue problems. Towards the end, an additional technique called preconditioning is discussed as a means to improve the convergence speed of the Jacobi-Davidson iteration.

3.2.1. The Davidson algorithm

Before quadratic eigenvalue problems, let's first consider a standard eigenvalue problem

$$A\mathbf{x} = \lambda \mathbf{x}. \tag{3.7}$$

We Let $\mathbf{v}_1, \cdots, \mathbf{v}_m$ be a set of orthonormal vectors which spans the search space $\mathcal{R}(V_m)$ with $V_m = [\mathbf{v}_1, \cdots, \mathbf{v}_m]$. In the Davidson algorithm we desire some vectors $\mathbf{s} \in \mathbb{C}^m$ satisfying

$$AV_m\mathbf{s} - \theta V_m\mathbf{s} \perp \mathbf{v}_1, \cdots, \mathbf{v}_m. \tag{3.8}$$

(3.8) is called the Galerkin condition, leading to a smaller eigenvalue problem

$$V_m^* A V_m \mathbf{s} = \theta V_m^* V_m \mathbf{s}. \tag{3.9}$$

The solutions to Eq. (3.9), the eigenpairs $(\theta_j^{(m)}, \mathbf{s}_j^{(m)}), j = 1, \cdots, m$, are called Ritz pairs.

The Ritz value $\theta_j^{(m)}$, along with the corresponding Ritz vector $\mathbf{u}_j = V_m\mathbf{s}_j^{(m)}$, can approximate the true eigenpairs of Eq.(3.7). Usually, we are looking for the largest or smallest eigenvalue in which case $j = 1$ or $j = m$ respectively. To evaluate how well the approximation is, compute the residual

$$\mathbf{r}_j = A\mathbf{u}_j - \theta_j\mathbf{u}_j. \tag{3.10}$$

We always pursue a smaller residual norm $\|\mathbf{r}_j\|$. Expanding the dimension of the search space is one way. To determine the basis vector used for expanding V_m, one can compute a vector $\mathbf{t}$ from

$$(D_A - \theta_j I)\mathbf{t} = \mathbf{r}_j, \tag{3.11}$$

where D_A is the diagonal of the matrix A. V_{m+1} is obtained by orthogonalizing $\mathbf{t}$ against V_m. The Davidson algorithm is given by following algorithm 1.

Algorithm 1 The Davidson algorithm

Input: initial vector $\mathbf{v}_1$ with $\|\mathbf{v}_1\| = 1$, $V_1 = [\mathbf{v}_1]$
for $j = 1, 2, \cdots$ **do**
  $B = V_j^* A V_j$
  Compute the largest eigenvalue θ of B
  and the corresponding eigenvector $\mathbf{s}$ with $\|\mathbf{s}\| = 1$
  $\mathbf{u} = V_j\mathbf{s}$
  $\mathbf{r} = A\mathbf{u} - \theta\mathbf{u}$
  $\mathbf{t} = (D_A - \theta I)^{-1}\mathbf{r}$
  $\mathbf{t} = \mathbf{t} - V_j V_j^*\mathbf{t}$
  $\mathbf{v}_{j+1} = \mathbf{t}/\|\mathbf{t}\|$
  $V_{j+1} = [V_j, \mathbf{v}_{j+1}]$
end for

3.2.2. The Jacobi orthogonal component correction

The Davidson algorithm works very well in searching dominant eigenvalues of diagonally dominant matrices. In addition to it, Jacobi gave a method for improving the eigenvector approximations. In [26], an iterative projection method was developed by combining the Davidson method with the Jacobi improvement, by Sleijpen and van der Vorst. Here we provide a concise explanation of the fundamental concept of the Jacobi component within the Jacobi-Davidson algorithm.

We again denote an approximation to the real eigenvector $\mathbf{x}$ of A by $\mathbf{u}_j$ and let θ_j be the Ritz value corresponding $\mathbf{u}_j$. We improve $\mathbf{u}_j$ by a vector $\mathbf{t}$ orthogonal to $\mathbf{u}_j$ such that

$$A(\mathbf{u}_j + \mathbf{t}) = \lambda(\mathbf{u}_j + \mathbf{t}). \tag{3.12}$$

(3.12) is called the Jacobi orthogonal component correction. If $\|\mathbf{u}_j\| = 1$ and Eq.(3.12) can be split into two parts:

$$\text{parallel to } \mathbf{u}_j\text{: } \mathbf{u}_j\mathbf{u}_j^* A(\mathbf{u}_j + \mathbf{t}) = \lambda\mathbf{u}_j\mathbf{u}_j^*(\mathbf{u}_j + \mathbf{t}), \tag{3.13}$$

and

$$\text{orthogonal to } \mathbf{u}_j\text{: } (I - \mathbf{u}_j\mathbf{u}_j^*)A(\mathbf{u}_j + \mathbf{t}) = \lambda(I - \mathbf{u}_j\mathbf{u}_j^*)(\mathbf{u}_j + \mathbf{t}). \tag{3.14}$$

Given $\theta_j = \mathbf{u}_j^* A\mathbf{u}_j$ and $\mathbf{u}_j^*\mathbf{u}_j = 1$ one can rewrite Eq.(3.14) as

$$\begin{aligned}(I - \mathbf{u}_j\mathbf{u}_j^*)(A\mathbf{t} - \lambda\mathbf{t}) &= (I - \mathbf{u}_j\mathbf{u}_j^*)(-A\mathbf{u}_j + \lambda\mathbf{u}_j) \\ &= -(I - \mathbf{u}_j\mathbf{u}_j^*)A\mathbf{u}_j = -(A - \theta_j I)\mathbf{u}_j =: -\mathbf{r}_j.\end{aligned}$$

λ, as the real eigenvalue of A, is of course unknown, so we replace λ by θ_j. As $(I - \mathbf{u}_j\mathbf{u}_j^*)\mathbf{t} = \mathbf{t}$, we can derive the Jacobi-Davidson correction equation

$$(I - \mathbf{u}_j\mathbf{u}_j^*)(A - \theta_j I)(I - \mathbf{u}_j\mathbf{u}_j^*)\mathbf{t} = -\mathbf{r}_j = -(A - \theta_j I)\mathbf{u}_j,\ t \perp \mathbf{u}_j. \tag{3.15}$$

The correction equation (3.15) is by convention solved by iterative solvers such as the generalized minimal residual method (GMRES) usually in low accuracy. Once we solve (3.15), we derive **t** as the improvement of the Ritz pair $(\theta_j, \mathbf{u}_j)$. In the Jacobi-Davidson method, the search space V_m which yields the Ritz pair is expanded by **t**, the solution of the correction equation. The next Ritz pair is determined by the expanded search space.

Algorithm 2 The Jacobi-Davidson algorithm computing a single eigenpair of A closest to a target value τ

Input: initial vector **t**
$V_0 = [\,]$, $V_0^A = [\,]$, $m = 0$
while stopping criterion not met **do**
  for $i = 1, \cdots, m-1$ **do**
    $\mathbf{t} = \mathbf{t} - (\mathbf{v}_i^*\mathbf{t})\mathbf{v}_i$ ▷ orthogonalization
  end for
  $\mathbf{v}_m = \mathbf{t}/\|\mathbf{t}\|$, $\mathbf{v}_m^A = A\mathbf{v}_m$, $V_m = [V_{m-1}, \mathbf{v}_m]$, $V_m^A = [V_{m-1}^A, \mathbf{v}_m^A]$
  for $i = 1, \cdots, m$ **do**
    $M_{i,m} = \mathbf{v}_i^*\mathbf{v}_m^A$, $M_{m,i} = \mathbf{v}_m^*\mathbf{v}_i^A$ ▷ $M = V_m^* A V_m$
  end for
  Compute the eigenvalue θ of M which is closest to the target τ
  Compute the corresponding eigenvector **s**: $M\mathbf{s} = \theta\mathbf{s}$, $\|\mathbf{s}\| = 1$ ▷ Rayleigh-Ritz
  $\mathbf{u} = V_m\mathbf{s}$, $\mathbf{u}^A = V_m^A\mathbf{s}$, $\mathbf{r} = \mathbf{u}^A - \theta\mathbf{u}$
  if $\|\mathbf{r}\| <$tol **then**
    return $\lambda = \theta$, $\mathbf{x} = \mathbf{u}$
  end if
  if $m = m_{max}$ **then** ▷ restart
    $V_{m_{min}} = V_m S_{:,1:m_{min}}$, where $S_{:,1:m_{min}}$ is composed of m_{min} eigenvectors computed in Rayleigh-Ritz step whose corresponding eigenvalues are closest to the target τ
  end if
  (Approximately) solve the correction equation for **t**
  $(I - \mathbf{u}\mathbf{u}^*)(A - \theta I)(I - \mathbf{u}\mathbf{u}^*)\mathbf{t} = -\mathbf{r}$, $\mathbf{t} \perp \mathbf{u}$
end while

Algorithm 2 gives a basic framework of the Jacobi-Davidson algorithm. Note that in algorithm 2 we introduce an operation called restart. Since the dimension of the search space V_m is larger and larger with iterations, it is necessary to limit the dimension of V_m, otherwise, the consumption of computation and memory resources could be extremely huge after many iterations. As long as the dimension m reaches m_{max}, we replace V_m with a matrix composed of m_{min} Ritz vectors corresponding to the Ritz values closest to the target τ.

Till now the whole Jacobi-Davidson process we've discussed can only compute a single eigenpair of a standard eigenvalue problem. Different from Arnoldi, Jacobi-Davidson can converge very rapidly to an eigenvalue closest to the target but on the other hand, can only find one eigenpair at a time. However, one is often interested not only in one but in several eigenpairs. A popular solution is to render

converged eigenvectors harmless by deflation. To implement the deflation, we need to work with the Schur decomposition.

Let's consider an orthonormal basis $\mathbf{v}_1, \cdots, \mathbf{v}_m$ of the current search space V_m. Suppose there is a matrix C given by

$$C = V_m^* A V_m,$$

and the Schur factorization of C is

$$CU = US, \tag{3.16}$$

where U is an orthonormal matrix and S is an upper triangular matrix. By reordering S such that $|s_{ii} - \tau|$ is non-increasing, the first diagonal entry of S represents the approximate eigenvalue closest to the target τ. And the corresponding columns of $V_m U$ span an approximation to the invariant subspace of A corresponding to these eigenvectors. The decomposition (3.16), $A(V_m U) = (V_m U)S$, can be used in a restart if one discards the columns of S and $V_m U$ corresponding to the undesired s_{ii}.

Now let's work on expanding the search space. Assume that a partial Schur form $AV_m = V_m R_m$ is known which we want to complement by a new column $\mathbf{v}$ such that

$$A \begin{bmatrix} V_m & \mathbf{v} \end{bmatrix} = \begin{bmatrix} V_m & \mathbf{v} \end{bmatrix} \begin{bmatrix} R_m & \mathbf{s} \\ 0 & \lambda \end{bmatrix}, \quad V_m \perp \mathbf{v}, \tag{3.17}$$

which is equivalent with

$$\begin{aligned} &AV_m = V_m R_m, \\ &A\mathbf{v} = V_m \mathbf{s} + \lambda \mathbf{v}, \quad V_m^* \mathbf{v} = 0. \end{aligned} \tag{3.18}$$

Multiplying the second equation of (3.18) with V_m^* yields

$$V_m^* A \mathbf{v} = V_m^* V_m \mathbf{s} + V_m^* \lambda \mathbf{v} = \mathbf{s}. \tag{3.19}$$

Insert Eq.(3.19) into the second equation of (3.18), we get

$$A\mathbf{v} = V_m V_m^* A \mathbf{v} + \lambda \mathbf{v}, \tag{3.20}$$

that is

$$(A - \lambda I)\mathbf{v} = V_m V_m^* A \mathbf{v}. \tag{3.21}$$

From $V_m^* \mathbf{v} = 0$ we can write

$$(A - \lambda I) V_m V_m^* \mathbf{v} = 0, \tag{3.22}$$

which leads to

$$(A - \lambda I)(I - V_m V_m^*)\mathbf{v} = V_m V_m^* A \mathbf{v}. \tag{3.23}$$

From Eq.(3.23), we can obtain

$$(I - V_m V_m^*)(A - \lambda I)(I - V_m V_m^*)\mathbf{v} = 0. \tag{3.24}$$

Eq.(3.24) demonstrates that the new pair (v, λ) is an eigenpair of

$$\tilde{A} = (I - V_m V_m^*) A (I - V_m V_m^*), \tag{3.25}$$

which can be determined by the Jacobi-Davidson algorithm again. With the deflation (3.25), the correction equation we need to solve for $\mathbf{t}$ is adapted to

$$(I - \mathbf{u}\mathbf{u}^*)(I - V_m V_m^*)(A - \theta I)(I - V_m V_m^*)(I - \mathbf{u}\mathbf{u}^*)\mathbf{t} = -\mathbf{r}, \tag{3.26}$$

where $(\theta, \mathbf{u})$ is the current Ritz pair.

So far we have gone through almost all necessary ingredients of the Jacobi-Davidson method for our research. The algorithm scheme discussed above is the foundation of other variants aiming for different kinds of eigenvalue problems. For generalized eigenvalue problems, we can apply one of the variants called the JDQZ algorithm which computes a partial QZ decomposition through the JD process. More complex nonlinear problems such as quadratic eigenvalue problems can also be addressed by means of a variant of the JD method. In the following subsections, we will introduce these variants separately.

3.2.3. JDQZ algorithm for generalized eigenvalue problems

There is a variant of Jacobi-Davidson called JDQZ [2, 11] that computes a partial QZ decomposition of the stencil (A, B) for the generalized eigenvalue problem

$$A\mathbf{x} = \lambda B\mathbf{x}. \tag{3.27}$$

Let $\lambda = \alpha/\beta$ for some α and β. The generalized eigenvalue equation (3.27) is equivalent to

$$(\beta A - \alpha B)\mathbf{x} = 0, \tag{3.28}$$

where we can denote the eigenvalue of the matrix pair (A, B) as a pair of (α, β). Eq.(3.28) emphasizes the symmetry of the roles of A and B.

A partial generalized Schur form (sometimes called QZ decomposition) [13] of dimension k for a matrix pair (A, B) of size n is defined by the factorization:

$$AQ_k = Z_k R_k^A, \quad BQ_k = Z_k R_k^B, \tag{3.29}$$

where Q_k and Z_k are unitary $n \times k$ matrices (the complex analogue of orthogonal matrices) and R_k^A and R_k^B are upper triangular $k \times k$ matrices. A column $\mathbf{q}_i$ of Q_k is referred to as a generalized Schur vector, and we refer to a pair $((\alpha_i, \beta_i), \mathbf{q}_i)$, with $(\alpha_i, \beta_i) = (R_k^A(i,i), R_k^B(i,i))$ as a generalized Schur pair. It follows that if $((\alpha, \beta), \mathbf{y})$ is generalized eigenpair of (R_k^A, R_k^B), then $((\alpha, \beta), Q_k\mathbf{y})$ is a generalized eigenpair of (A, B).

From the partial Schur decomposition (3.29) we can derive

$$\beta_i A\mathbf{q}_i - \alpha_i B\mathbf{q}_i \perp \mathbf{z}_i.$$

After setting a Petrov-Galerkin condition to construct reduced systems, we choose the approximate eigenvector $\mathbf{u}$ for each iteration from a j-dimensional search subspace $span(V_j) = span\{\mathbf{v}_1, \cdots, \mathbf{v}_j\}$. With the approximate eigenvector $\mathbf{u}$ and generalized pair (η, ζ), the residual $\eta A\mathbf{u} - \zeta B\mathbf{u}$ is required to be orthogonal to some other well-chosen test subspace $span(W_j) = span\{\mathbf{w}_1, \cdots, \mathbf{w}_j\}$,

$$\eta A\mathbf{u} - \zeta B\mathbf{u} \perp span(W_j). \tag{3.30}$$

This requirement leads to a projected generalized $j \times j$ eigenproblem

$$(\eta W_j^* AV_j - \zeta W_j^* BV_j)\mathbf{s} = 0. \tag{3.31}$$

The j-dimensional pencil $\eta W_j^* AV_j - \zeta W_j^* BV_j$ can be reduced by the QZ algorithm to generalized Schur form, which leads to orthogonal $j \times j$ matrices S^R and S^L and upper triangular $j \times j$ matrices T^A and T^B such that

$$\begin{aligned} (S^L)^*(W_j^* AV_j)S^R &= T^A, \\ (S^L)^*(W_j^* BV_j)S^R &= T^B. \end{aligned} \tag{3.32}$$

This decomposition can be reordered such that the first column of S^R and the (1,1)-entries of T^A and T^B represent the wanted Petrov solution. With $\mathbf{s} := \mathbf{s}_1^R := S^R\mathbf{e}_1$ and $\zeta := T_{1,1}^A, \eta := T_{1,1}^B$, the Petrov vector is defined as

$$\mathbf{u} := V_j\mathbf{s} = V_j\mathbf{s}_1^R \tag{3.33}$$

for the associated generalized Petrov value (ζ, η). Similarly, we can define a left Petrov vector as

$$\mathbf{p} := W_j \mathbf{s}_1^L, \quad \mathbf{s}_1^L := S^L \mathbf{e}_1. \tag{3.34}$$

If V_j and W_j are unitary, then $\|\mathbf{s}^R\|_2 = \|\mathbf{s}^L\|_2 = 1$ implies $\|\mathbf{u}\|_2 = 1$. With the decomposition in (3.32), we construct an approximate partial generalized Schur form: $V_j S^R$ approximates a Q_k, and $W_j S^L$ approximates the associated Z_j.

Now the choice of the test space W_j is still not clear to us. Choose W_j such that $span(W_j)$ coincides with $span(\nu_0 A V_j + \mu_0 B V_j)$ for some proper ν_0 and μ_0. With the weights ν_0 and μ_0 we can influence the convergence of the Petrov values. If we want eigenpair approximations for eigenvalues λ close to a target τ, then the choice

$$\nu_0 = \frac{1}{\sqrt{1+|\tau|^2}}, \quad \mu_0 = -\tau\nu_0$$

is quite effective, especially when we desire eigenvalues in the interior of the spectrum of $A - \lambda B$. The Jacobi-Davidson correction for the component $\mathbf{t} \perp \mathbf{u}$ for the pencil $\eta A - \zeta B$ is given by

$$(I - \mathbf{p}\mathbf{p}^*)(\eta A - \zeta B)(I - \mathbf{p}\mathbf{p}^*)\mathbf{t} = -\mathbf{r}, \quad \mathbf{r} := (\eta A - \zeta B)\mathbf{u}. \tag{3.35}$$

Similar to the original Jacobi-Davidson algorithm, the correction equation is typically solved with a preconditioned iterative solver to obtain an approximate vector $\mathbf{t}$. This vector is then utilized to expand V_j, while $\nu_0 A\mathbf{v} + \mu_0 B\mathbf{v}$ is used to expand W_j. And a modified Gram-Schmidt orthogonalization process is applied to make sure the new columns are all orthonormal to the current basis. JDQZ algorithm also possesses a restart process. Similar to the JD process, in order to reduce the dimension of the subspaces to $j_{min} < j$, we impose such restart operation:

$$V = [V\mathbf{s}_1^R, \cdots, V\mathbf{s}_{j_{min}}^R], \quad W = [W\mathbf{s}_1^L, \cdots, W\mathbf{s}_{j_{min}}^L],$$

where $V\mathbf{s}_1^R, \cdots, V\mathbf{s}_{j_{min}}^R$ are the j_{min} most promising Petrov vectors.

Besides restart, in JDQZ the converged Petrov (analogy to Ritz for JD) can also be deflated. To obtain the partial generalized Schur form, we proceed as follows. Suppose we have already computed the partial generalized Schur form $AQ_{k-1} = Z_{k-1}R_{k-1}^A$ and $BQ_{k-1} = Z_{k-1}R_{k-1}^B$. We expand this partial generalized Schur form with the new right Schur vector u and the left Schur vector $\mathbf{p}$ to

$$A\begin{bmatrix}Q_{k-1}\mathbf{u}\end{bmatrix} = \begin{bmatrix}Z_{k-1}\mathbf{p}\end{bmatrix}\begin{bmatrix}R_{k-1}^A & \mathbf{a} \\ 0 & \alpha\end{bmatrix} \tag{3.36}$$

and

$$A\begin{bmatrix}Q_{k-1}\mathbf{u}\end{bmatrix} = \begin{bmatrix}Z_{k-1}\mathbf{p}\end{bmatrix}\begin{bmatrix}R_{k-1}^B & \mathbf{b} \\ 0 & \beta\end{bmatrix}. \tag{3.37}$$

The new generalized Schur pair $((\alpha, \beta), \mathbf{u})$ satisfies

$$Q_{k-1}^*\mathbf{u} = 0, \quad (\beta A - \alpha B)\mathbf{u} - Z_{k-1}(\beta\mathbf{a} - \alpha\mathbf{b}) = 0.$$

Algorithm 3 illustrates the process of the Jacobi-Davidson algorithm computing a partial QZ decomposition for a general matrix pencil (A, B). In the algorithm the generalized Schur pairs $((\alpha, \beta), \mathbf{u})$, for which the ratio β/α is closest to a target τ, is computed. The vectors $\mathbf{a}$ and $\mathbf{b}$ can be computed from

$$\mathbf{a} = Z_{k-1}^* A\mathbf{u}, \quad \mathbf{b} = Z_{k-1}^* B\mathbf{u}. \tag{3.38}$$

The generalized Schur pair $((\alpha, \beta), \mathbf{u})$ is an eigenpair of the deflated matrix pair

$$((I - Z_{k-1}Z_{k-1}^*)A(I - Q_{k-1}Q_{k-1}^*), (I - Z_{k-1}Z_{k-1}^*)B(I - Q_{k-1}Q_{k-1}^*)). \tag{3.39}$$

This eigenproblem can be solved again with the JDQZ algorithm. With the deflated matrix pair (3.39), we simplify the computation of the interaction matrices M^A and M^B:

$$\begin{aligned} M^A &= W^*(I - Z_{k-1}Z_{k-1}^*)A(I - Q_{k-1}Q_{k-1}^*)V = W^*AV, \\ M^B &= W^*(I - Z_{k-1}Z_{k-1}^*)B(I - Q_{k-1}Q_{k-1}^*)V = W^*BV. \end{aligned} \tag{3.40}$$

Algorithm 3 The Jacobi-Davidson QZ algorithm computing k_{max} eigenvalues closest to a target value τ for the generalized eigenvalue problem

This algorithm computes k_{max} eigenvalues of $\alpha A\mathbf{x} = \beta B\mathbf{x}$ closest to the target τ.
Input: initial vector $\mathbf{v}_0$
$\mathbf{t} = \mathbf{v}_0$, $k = 0$, $\nu_0 = 1/\sqrt{1+\tau^2}$, $\mu_0 = -\tau\mu_0$, $m = 0$
$Q = [\,]$, $Z = [\,]$, $S = [\,]$, $T = [\,]$
while $k < k_{max}$ **do**
  $\mathbf{t} = \mathbf{t} - V_m V_m^*\mathbf{t}$ ▷ Orthogonalization
  $m = m+1$, $\mathbf{v}_m = \mathbf{t}/\|\mathbf{t}\|$, $\mathbf{v}_m^A = A\mathbf{v}_m$, $\mathbf{v}_m^B = B\mathbf{v}_m$, $\mathbf{w} = \nu_0\mathbf{v}_m^A + \mu_0\mathbf{v}_m^B$
  $\mathbf{w} = \mathbf{w} - Z_k Z_k^*\mathbf{w}$ ▷ Orthogonalization
  $\mathbf{w} = \mathbf{w} - W_{m-1}W_{m-1}^*\mathbf{w}$ ▷ Orthogonalization
  $\mathbf{w}_m = \mathbf{w}/\|\mathbf{w}\|$
  $M^A = \begin{bmatrix} M^A & W_{m-1}^*\mathbf{v}_m^A \\ \mathbf{w}_m^* V_{m-1}^A & \mathbf{w}_m^*\mathbf{v}_m^A \end{bmatrix}$, $M^B = \begin{bmatrix} M^B & W_{m-1}^*\mathbf{v}_m^B \\ \mathbf{w}_m^* V_{m-1}^B & \mathbf{w}_m^*\mathbf{v}_m^B \end{bmatrix}$
  Compute the QZ decomposition $M^A S^R = S^L T^A$ and $M^B S^R = S^L T^B$ such that $|T_{i,i}^A/T_{i,i}^B - \tau| \leq |T_{i+1,i+1}^A/T_{i+1,i+1}^B - \tau|$ ▷ Rayleigh-Ritz
  $\mathbf{u} = V\mathbf{s}_1^R$, $\mathbf{p} = W_j\mathbf{s}_1^L$, $\mathbf{u}^A = V^A\mathbf{s}_1^R$, $\mathbf{u}^B = V^B\mathbf{s}_1^R$, $\zeta = T_{1,1}^A$, $\eta = T_{1,1}^B$
  $\mathbf{r} = \eta\mathbf{u}^A - \zeta\mathbf{u}^B$, $\tilde{\mathbf{a}} = Z^*\mathbf{u}^A$, $\tilde{\mathbf{b}} = Z^*\mathbf{u}^B$, $\tilde{\mathbf{r}} = \mathbf{r} - Z(\eta\tilde{\mathbf{a}} - \zeta\tilde{\mathbf{b}})$
  while $\|\tilde{\mathbf{r}}\| < \epsilon$ **do**
    $R^A = \begin{bmatrix} R^A & \tilde{\mathbf{a}} \\ 0^\top & \zeta \end{bmatrix}$, $R^B = \begin{bmatrix} R^B & \tilde{\mathbf{b}} \\ 0^\top & \eta \end{bmatrix}$
    $Q = [Q, \mathbf{u}]$, $Z = [Z, \mathbf{p}]$, $k = k+1$
    if $k = k_{max}$ **then**
      return (Q, Z, R^A, R^B)
    end if
    $m = m-1$
    for $i = 1, \cdots, m$ **do**
      $\mathbf{v}_i = V\mathbf{s}_{i+1}^R$, $\mathbf{v}_i^A = V^A\mathbf{s}_{i+1}^R$, $\mathbf{v}_i^B = V^B\mathbf{s}_{i+1}^R$, $\mathbf{w}_i = W\mathbf{s}_{i+1}^L$, $\mathbf{s}_i^R = \mathbf{s}_i^L = \mathbf{e}_i$
    end for
    $\mathbf{u} = \mathbf{u}_1$, $\mathbf{p} = \mathbf{w}_1$, $\mathbf{u}^A = \mathbf{v}_1^A$, $\mathbf{u}^B = \mathbf{v}_1^b$, $\zeta T_{1,1}^A$, $\eta = T_{1,1}^B$
    $\mathbf{r} = \eta\mathbf{u}^A - \zeta\mathbf{u}^B$, $\tilde{\mathbf{a}} = Z^*\mathbf{u}^A$, $\tilde{b} = Z^*\mathbf{u}^B$, $\tilde{\mathbf{r}} = \mathbf{r} - Z(\eta\tilde{\mathbf{a}} - \zeta\tilde{\mathbf{b}})$
  end while
  if $m \geq m_{max}$ **then**
    for $i = 2, \cdots, m_{min}$ **do**
      $\mathbf{v}_i = V\mathbf{s}_i^R$, $\mathbf{v}_i^A = V^A\mathbf{s}_i^R$, $\mathbf{v}_i^B = V^B\mathbf{s}_i^R$, $\mathbf{w}_i = W\mathbf{s}_i^L$
    end for
    $\mathbf{v}_1 = \mathbf{u}$, $\mathbf{v}_1^A = \mathbf{u}^A$, $\mathbf{v}_1^B = \mathbf{v}^B$, $\mathbf{w}_1 = \mathbf{p}$, $m = m_{min}$
  end if
  $\tilde{Q} = [Q, \mathbf{u}]$, $\tilde{Z} = [Z, \mathbf{p}]$
  Solve the correction equation for t: $(I - \tilde{Z}\tilde{Z}^*)(\eta A - \zeta B)(I - \tilde{Q}\tilde{Q}^*)\mathbf{t} = -\mathbf{r}$
end while

3.2.4. JD algorithm for quadratic eigenvalue problems

Recall that the task of JDQZ in our research is to solve the generalized eigenvalue problem twice bigger in size than the quadratic problem. Keeping this point in mind, one may ask if there is an alternative way to solve the QEP directly without reducing the quadratic structure of the problem. Again, the Jacobi-Davidson method provides us with such an opportunity. Here we introduce a variant of the JD algorithm proposed in [12, 25] designed to search several eigenpairs of the quadratic eigenvalue problem.

In this thesis project, we will follow the idea of algorithm 4 to directly solve the quadratic eigenvalue problem (2.27) by means of the JD method. Let us make a few comments on algorithm 4. As a starting vector expanding the search space, $\mathbf{t}$ is at first normalized in M-norm, i.e. $\mathbf{t}^* M\mathbf{t} = 1$. If a proper approximation of the desired eigenvector is known, we should take it as the initial vector. Otherwise, we can make $\mathbf{t}$ a constant vector, of which all coefficients are equal to one. In the orthogonalization

step, rgs denotes repeated Gram–Schmidt orthogonalization which can stably compute an orthonormal basis. Then in the Rayleigh-Ritz step, we compute the eigenpairs of the projected small QEP, $\theta^2\mathbf{s} + \theta H_C\mathbf{s} + H_K\mathbf{s} = 0$. We can apply the linearization approach discussed in Section 3.1 to solve the equation. The size of the equation is equal to the dimension of the search space V_m. As the size is not big, we can solve the projected QEP quickly. We usually regard the eigenpair with the smallest eigenvalue or the nearest eigenvalue to the target as the most promising solution and take it as an approximation to the real eigenpair.

Algorithm 4 The Jacobi-Davidson algorithm computing several eigenpairs of the quadratic eigenvalue problem $\lambda^2 M\mathbf{p} + \lambda C\mathbf{p} + K\mathbf{p} = 0$ closest to a target value τ

Input: initial vector t. Normalize **t** in M-norm
$V_0 = [\,]$, $m = 0$
while number of eigenvalues desired not met **do**
  $m = m + 1$, $V_m = rgs(V_{m-1}, \mathbf{t})$ ▷ orthogonalization
  Compute KV_m, CV_m, MV_m
  Compute the projected matrices $H_K = V_m^* KV_m, H_C = V_m^* CV_m$
  Compute approximation to the desired eigenvalue θ and corresponding eigenvector
  s of the projected problem $\theta^2\mathbf{s} + \theta H_C\mathbf{s} + H_K\mathbf{s} = 0, \|\mathbf{s}\| = 1$ ▷ Rayleigh-Ritz
  $\mathbf{u} = V_m\mathbf{s}$
  $\mathbf{w} = 2\theta M\mathbf{u} + C\mathbf{u}$, $\mathbf{r} = \theta^2 M\mathbf{u} + \theta C\mathbf{u} + K\mathbf{u}$
  if $\|\mathbf{r}\| <$tol **then**
    Accept the approximate eigenpair $(\theta, \mathbf{u})$
    Choose an approximation to the next eigenpair
    Compute the residual **r**
  end if
  if $m = m_{max}$ **then** ▷ restart
    $V_{m_{min}} = V_m S_{:,1:m_{min}}$, where $S_{:,1:m_{min}}$ is composed of m_{min} eigenvectors computed in Rayleigh-Ritz step whose corresponding eigenvalues are closest to the target τ
  end if
  (Approximately) solve the correction equation for t (possibly with conditioner)
  $(I - (\mathbf{w}\mathbf{u}^*)/(\mathbf{u}^*\mathbf{w}))(\theta^2 M + \theta C + K)(I - (\mathbf{u}\mathbf{u}^*)/(\mathbf{u}^*\mathbf{u}))\mathbf{t} = -\mathbf{r}$, $\mathbf{t} \perp \mathbf{u}$ ▷ correction
end while

As the subspace grows, solving the projected eigenvalue problem incurs higher storage and computational cost, necessitating periodic restarts of the algorithm. The restarted search space consists of the most promising solutions of the projected eigenvalue problem, i.e. the eigenvectors whose eigenvalues are closest to the target value. H. Voss (2007) [31] states that a restart destroys information on the eigenvectors, particularly on the one the method aims at. To mitigate potential information loss, [31] proposes a strategy of restarting only when a single eigenvector has converged while computing several eigenpairs. However, in our algorithms, we maintain the approach of restarting regardless of whether any eigenpair has converged. This is because it is not guaranteed that the algorithm can converge to an eigenpair before running out of memory. This decision prioritizes reducing each iteration's computational burden and memory consumption over achieving faster convergence.

Once an eigenpair converges, we immediately choose another eigenpair as the next approximation. A crucial problem the Jacobi-Davidson method faces when approximating multiple eigenvalues is how to prevent the method from repeatedly converging to the same eigenvalue. Computing more than one eigenvalue for large polynomial eigenproblems is a much harder task than for standard eigenvalue problems. To avoid detecting eigenpairs that have been found by previous iterations, one effective approach is to lock the detected eigenvectors within projectors, i.e. retain them in the search space. The term "locking" is employed to describe this technique later in the article. Later in the chapter dedicated to numerical results, we will demonstrate the effectiveness of "locking" in mitigating repetitive convergence. Moreover, an alternative approach involves implementing a deflation strategy proposed by [10]. Besides, one can filter the repeated eigenvalues by some special selection criterion [17].

3.2.5. Preconditioning

The Jacobi-Davidson algorithm has a big disadvantage in that one has to solve the correction equation in every iteration. The efficiency of solving the correction equation (3.15)/(3.35) determines the efficiency of searching good approximations of JD. Preconditioning of the correction equation can improve the convergence speed of GMRES.

In the expansion of the search space, it is ensured that the Newton iterate is contained in the expanded search space. Recall the correction equation of the quadratic Jacobi-Davidson algorithm

$$(I - \frac{\mathbf{w}\mathbf{u}^*}{\mathbf{u}^*\mathbf{w}})T(\theta)(I - \frac{\mathbf{u}\mathbf{u}^*}{\mathbf{u}^*\mathbf{u}})\mathbf{t} = -\mathbf{r} = -T(\theta)\mathbf{u},\ \mathbf{t} \perp \mathbf{u}, \tag{3.41}$$

where

$$T(\theta) = K + \theta C + \theta^2 M.$$

Given $\mathbf{t}^*\mathbf{u} = 0$, the correction equation can be written as

$$T(\theta)\mathbf{t} - \alpha\mathbf{w} = -\mathbf{r},\ \ \alpha = \frac{\mathbf{u}^*T(\theta)\mathbf{t}}{\mathbf{u}^*\mathbf{w}}.$$

By inserting $T(\theta)\mathbf{u} = \mathbf{r}$ into Eq. (3.42), we derive

$$\mathbf{t} = -\mathbf{u} + \alpha T(\theta)^{-1}\mathbf{w} = -\mathbf{u} + \alpha T(\theta)^{-1}T'(\theta)\mathbf{u}. \tag{3.42}$$

α is determined such that $\mathbf{t} \perp \mathbf{u}$. With this notation what we have to solve becomes $T(\theta)\tilde{\mathbf{t}} = T'(\theta)\mathbf{u}$ for some $\tilde{\mathbf{t}}$. From $\mathbf{u} = V\mathbf{s}$ and $\mathbf{t} \perp \mathbf{u}$ we can derive $\tilde{\mathbf{t}} = T(\theta)^{-1}T'(\theta)\mathbf{u} \in span\{V, \mathbf{t}\}$. Therefore, since in the linear case the new search space $span\{V, \mathbf{t}\}$ contains the vector obtained by one step of inverse iteration with shift θ and initial vector $\mathbf{u}$, we can expect quadratic convergence of the resulting iterative projection method if the correction equation is solved exactly [31].

However, when the problem size is large, it is infeasible to solve the correction equation exactly. As indicated in [26], the equation does not need to be solved accurately. Instead, some iterative methods, such as the generalized minimal residual method (GMRES), can be used to approximate the solution, but the performance of the Jacobi Davidson algorithm depends highly on the convergence of the GMRES correction solver. In principle, to improve the rate of convergence of the GMRES algorithm, it is useful to introduce a preconditioning matrix with a smaller condition number than the linear system which is often performed through a matrix-free method. Considering the restriction to the orthogonal complement of the current approximation $\mathbf{u}$, usually we choose the preconditioner of the form

$$\tilde{P} = (I - \frac{\mathbf{w}\mathbf{u}^*}{\mathbf{u}^*\mathbf{w}})P(I - \frac{\mathbf{u}\mathbf{u}^*}{\mathbf{u}^*\mathbf{u}}),\ \ P^{-1}T(\theta) \approx I. \tag{3.43}$$

In order to reduce the computational effort, P should possess easy invertibility. Consider a left preconditioner $\tilde{P}$, in every GMRES iteration we need to determine a vector $\mathbf{z} = \tilde{P}^{-1}\tilde{T}\mathbf{w}$ for some $\mathbf{w}$ where

$$\tilde{T} = (I - \frac{\mathbf{w}\mathbf{u}^*}{\mathbf{u}^*\mathbf{w}})T(\theta)(I - \frac{\mathbf{u}\mathbf{u}^*}{\mathbf{u}^*\mathbf{u}}).$$

Following $\mathbf{u}^*\mathbf{w} = 0$, $\tilde{T}\mathbf{w}$ is given by

$$\tilde{T}\mathbf{w} = (I - \frac{\mathbf{w}\mathbf{u}^*}{\mathbf{u}^*\mathbf{w}})T(\theta)(I - \frac{\mathbf{u}\mathbf{u}^*}{\mathbf{u}^*\mathbf{u}})\mathbf{w} = \mathbf{y},$$

where $\mathbf{y} = (I - (\mathbf{w}\mathbf{u}^*)/(\mathbf{u}^*\mathbf{w}))T(\theta)\mathbf{w}$. Computing $\mathbf{z} = \tilde{P}^{-1}\tilde{T}\mathbf{w}$ is equivalent to solving

$$\tilde{P}\mathbf{z} = \mathbf{y}. \tag{3.44}$$

As $\mathbf{z} \perp \mathbf{u}$, $\mathbf{z}$ has to satisfy

$$(I - \frac{\mathbf{w}\mathbf{u}^*}{\mathbf{u}^*\mathbf{w}})P\mathbf{z} = P\mathbf{z} - \frac{\mathbf{w}\mathbf{u}^*}{\mathbf{u}^*\mathbf{w}}P\mathbf{z} = \mathbf{y}, \tag{3.45}$$

The solution of Eq. (3.45) is given by

$$\mathbf{z} = P^{-1}\mathbf{y} - \alpha P^{-1}\mathbf{w}, \text{ where } \alpha = -\frac{\mathbf{u}^* P\mathbf{z}}{\mathbf{u}^*\mathbf{w}}. \tag{3.46}$$

With the constraint $\mathbf{z}^*\mathbf{u} = 0$, we can determine the scalar α by left-multiplying equation (3.46) with $\mathbf{u}^*$. Therefore,

$$\alpha = \frac{\mathbf{u}^* P^{-1}\mathbf{y}}{\mathbf{u}^* P^{-1}\mathbf{w}}.$$

Hence, in every step of the preconditioned GMRES we need to solve the linear system $P\tilde{\mathbf{y}} = \mathbf{y}$. And additionally we also have to solve the system $P\tilde{\mathbf{w}} = \mathbf{w}$ prior to the beginning of iteration. To conclude, the workflow of approximating solution of the correction equation through preconditioned GMRES is as follows:

1. Solve $P\tilde{\mathbf{w}} = \mathbf{w}$ for $\tilde{\mathbf{w}}$ and compute $\mu = \mathbf{u}^*\tilde{\mathbf{w}}$.
2. Solve $P\tilde{\mathbf{r}} = \mathbf{r}$ and set $\hat{\mathbf{r}} = \tilde{\mathbf{r}} - \frac{\mathbf{u}^*\tilde{\mathbf{r}}}{\mu}\tilde{\mathbf{w}}$.
3. Apply GMRES solver with initial vector $\mathbf{t}_0 = \mathbf{0}$ to solve $\tilde{P}^{-1}\tilde{T}\mathbf{t} = -\hat{\mathbf{r}}$. To perform preconditioning in a matrix-free way, in each step determine $\mathbf{z} = \tilde{P}^{-1}\tilde{T}\mathbf{p}$ for any vector $\mathbf{p}$ according to
   (a) $\mathbf{y} = (I - \frac{\mathbf{w}\mathbf{u}^*}{\mathbf{u}^*\mathbf{w}})T(\theta)\mathbf{p}$,
   (b) solve $P\tilde{\mathbf{y}} = \mathbf{y}$ for $\tilde{\mathbf{y}}$,
   (c) $\mathbf{z} = \tilde{\mathbf{y}} - \frac{\mathbf{u}^*\tilde{\mathbf{y}}}{\mu}\tilde{\mathbf{w}}$.

In order to solve the linear system $P\tilde{\mathbf{x}} = \mathbf{x}$ for $\tilde{\mathbf{x}}$ exactly, P should be easily invertible in principle, the choice of which depends on the problem T.

4 Parallelization

The linear algebra operations involved in the Jacobi-Davidson algorithm are highly suitable for parallelization. The motivation for parallelization is primarily driven by the computational demands arising from higher-dimensional problems. As three-body problems exhibit higher dimensions compared to two-body problems, the size and complexity of the problem increase significantly. Consequently, it becomes beneficial to take advantage of parallel computing on supercomputers. In the present chapter, we aim to showcase the intricate process of parallelizing the Jacobi-Davidson algorithm on supercomputers by explaining how we parallelize different linear algebra operations in the algorithm. We will exploit the parallelism of the tensor formulation of three-body problems to implement an efficient operator application.

4.1. Implementation of parallelization

In Chapter 3, we have introduced two approaches to solving the quadratic eigenvalue problems: (1) Linearization combined with some linear eigensolvers, such as Jacobi-Davidson QZ (Alg. 3). (2) The nonlinear Jacobi-Davidson method (Alg. 4). However, we have argued that linearization doubles the problem size and fails to preserve the tensor structure similar to the damping and stiffness matrices in the linearized pencil. Therefore, we choose to parallelize the nonlinear Jacobi-Davidson algorithm to solve the QEP arising from three-body problems.

The basic idea of parallelizing the algorithm involves dividing a computational task into smaller subtasks, distributing them among multiple processing units, and executing them concurrently. Parallelization aims to harness the collective processing power to solve the problem faster and more efficiently than a sequential execution on a single processing unit. The clustering of processing units may share the memory or use distributed memory. In our article, we will refer to the processing units using shared memory as threads and those using distributed memory as processes separately. To leverage the computational resource of multiple processes/threads, we need to utilize parallel computing libraries:

- Open Multi-Processing (OpenMP). OpenMP is an industry-standard API for shared-memory parallelism in multi-threaded programming. It allows developers to parallelize their code by adding directives, pragmas, and function calls that enable automatic thread creation and management. OpenMP simplifies parallel programming by providing a set of directives that indicate which parts of the code should be executed in parallel.

- Message Passing Interface (MPI). MPI is a library interface for message-passing parallel computing. It enables the development of parallel applications that run on distributed-memory systems, where multiple independent machines or processors are connected in a network. MPI supports various communication primitives, collective operations, and synchronization mechanisms. With MPI, programmers can create parallel applications by explicitly defining the communication patterns and data exchanges between different processes.

We have developed a multi-processor, multi-threaded C++ implementation to enable efficient parallelism for the three-body problem. In our program, multi-threaded computation is provided by OpenMP, and multi-processor computation is facilitated by MPI. The source code of the implementation is available in the GitHub repository at https://github.com/MengZhaonan1998/q_3body_wave_hpc.git. We employ the programming model of Single Program, Multiple Data (SPMD), a parallel programming paradigm where a single program is executed by multiple processing units simultaneously, with each processing unit operating on its own portion of the data. SPMD is commonly used in parallel computing to achieve parallelism and exploit the computational power of multiple processors or compute nodes. It leverages the inherent data-level parallelism in certain applications to improve performance by processing multiple data elements in parallel.

In SPMD programming, the same instruction is executed simultaneously on different data elements or separate domains. This approach is suitable for the Jacobi-Davidson method, where the same operation needs to be applied to a large amount of independent data, specifically vectors or matrices. More specifically, the primary focus for parallelization of the JD algorithm lies in parallelizing the linear algebra operations, such as the initialization of the vector (*init*), the addition of scalar-vector product (*axpby*), the scalar product (*dot*), matrix-vector multiplication (*matvec*), and so on. Following this idea, during the program design phase, we extract the linear algebra operations as a library, facilitating convenient parallelization of operations. The hierarchy of the whole program is illustrated by Fig. 4.1.

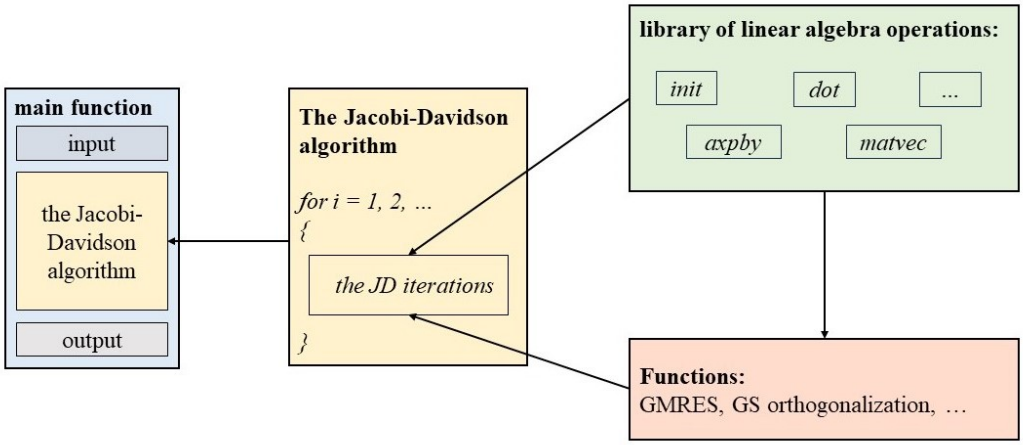


Figure 4.1: Architecture of the C++ implementation of the Jacobi-Davidson algorithm.

The arrow shown in Fig. 4.1 represents the dependencies between functions. We have a main function responsible for controlling the input arguments and output results, and it calls the Jacobi-Davidson function. The Jacobi-Davidson algorithm, in turn, calls several other functions, such as GMRES or GS orthogonalization. All these functions depend on a kernel library of linear algebra operations, which includes all the necessary operations that can potentially be parallelized. In this way, we initially implement a sequential program. Subsequently, we can harness the power of parallel computing by making modifications solely to the kernel library. By doing so, we can exploit parallelism without the need to alter the overall structure of the program. This strategy allows for an efficient transition from a sequential implementation to one that fully capitalizes on the benefits of parallel processing.

Implementing parallelization of those linear algebra operations in shared memory environments is relatively easy. For instance, to parallelize the loop of *init*, we simply add an OpenMP directive to enable parallel execution:

```
#pragma omp parallel for
  for (int i=0; i<n; i++)
     x[i] = value;
```

Same for *axpby*:

```
#pragma omp parallel for
  for (int i=0; i<n; i++)
     y[i] = (a * x[i] + b * y[i]);
```

When it comes to parallelism in distributed memory environments, the operations of *init* and *axpby* are both automatically parallel among processors. That is because there is typically no need for data communication among processors. Each processor can independently compute its portion of the operation using its local data.

In contrast, when dealing with the scalar product operation, which involves the need to gather and sum up the products from each processor/thread, an additional consideration called reduction arises. Specifically, in the scenario of *dot*, it is necessary to perform a reduction operation in both shared memory and distributed memory. To achieve this in shared memory, we can easily implement the reduction operation using OpenMP's *reduction* clause. On the other hand, in a distributed memory environment, MPI provides a collective communication routine known as *MPI_Allreduce*.

```
#pragma omp parallel for reduction(+:local_res)
  for (int i=0; i<n; i++)
     local_res += x[i]*y[i];
  MPI_Allreduce(&local_res, &global_res, 1, MPI_DOUBLE_COMPLEX, MPI_SUM, MPI_COMM_WORLD);
```

The most challenging aspect of parallelizing the JD method lies in effectively implementing the matvec operation, which involves the application of the stiffness matrix K, damping matrix C, and mass matrix M to arbitrary vectors. This *matvec* operation is at the core of the implementation of the Jacobi-Davidson algorithm, and its efficient parallelization is crucial for achieving significant speedup and scalability. In the last subsection of Chapter 2, we have shown that in the 1D three-body problem, M is a diagonal matrix, while K and C are sparse matrices possessing tensor structure. Therefore, the application of M can be implemented through a simple scalar-vector product. However, applying K and C is more intricate. In [27], a novel tensor product scheme is employed to minimize redundant computations during the operator application for computing bound states of three-body problems. The upcoming section will explore the feasibility of utilizing a similar tensor structure approach in our operator application.

4.2. Operator application for the 1D three-body problem

Section 2.2.2 has demonstrated that the application of the Kronecker product transforms our problem from a dense system to a sparse equation while preserving the tensor structure. The application of Kronecker products allows us to avoid storing every entry of the sparse matrix and instead store only the smaller dense matrices, which saves a significant amount of memory. In this section, we leverage the tensor product to develop an efficient implementation of the operator application for the one-dimensional three-body problem.

4.2.1. Tensor product scheme

The Kronecker product is utilized to formulate the three-body problem, resulting in sparse a stiffness matrix $K^{(3b)}$ and a damping matrix $C^{(3b)}$ preserving a tensor structure. Based on the formulation of the three-body problem discussed previously, our linear operator can be written in a general form

$$T_{V,a_1,a_2} = a_1(B_1 \otimes \mathbb{I}_2) + a_2(\mathbb{I}_1 \otimes B_2) + V, \tag{4.1}$$

where B_1 and B_2 correspond to the linear operators in the x and y coordinates, respectively, and V is a diagonal matrix standing for potential $(K^{(3b)})$ or zero $(C^{(3b)})$. For example, in the context of the stiffness matrix, $B_1 = K_x$, $B_2 = K_y$, and V is a diagonal matrix where its diagonal entries represent the potential. The damping matrix can be obtained analogously, but note that there are no potential terms V.

Thies et al. [27] propose an efficient implementation that utilizes dense matrix-matrix products to apply the operator T_{V,a_1,a_2} to a vector w. This approach is highly recommended for us to adopt and follow, as it offers a beneficial strategy for improving computational efficiency and saving memory. Let

$B_1 \in \mathbb{R}^{N_1 \times N_1}$, $B_2 \in \mathbb{R}^{N_2 \times N_2}$, $w \in \mathbb{R}^{N_1 N_2}$, and $W = \text{reshape}(w, N_2, N_1)$ denote the interpretation of w as an $N_2 \times N_1$ matrix. We can then derive

$$T_{V,a_1,a_2} \cdot w = \text{reshape}(a_2 B_2 \cdot W + a_1 W \cdot B_1^T, N_1 N_2, 1) + V \cdot w, \tag{4.2}$$

where the reshape operation is used to interpret the resulting $N_2 \times N_1$ matrix as a vector of length $N_1 N_2$. It is worth noting that reshape does not involve any data movement. It is just a re-interpretation of a vector as a matrix stored in column-major ordering and vice versa. If $N_x = N_y = n$, by adopting a new approach, the storage requirement of the Hamiltonian operator in 1D is significantly reduced to $O(n^2)$, which is a substantial improvement compared to the $O(n^3)$ storage requirement when utilizing a sparse matrix format.

To parallelize the application of the linear operators, we use a column-wise distribution of W from Eq. (4.2) among the processes, while the dense matrices B_1 and B_2 are replicated on all processes. In our codes, we first compute $a_2 B_2 \cdot W$, followed by $a_1 W \cdot B_1^T$. Note that during the computation of $a_1 W \cdot B_1^T$, each processor has to access the entries of W that are stored in other processors, resulting in the need for data communication between the processors. We can avoid the idle time caused by communication through the non-blocking operation. The computation steps carried out by each processor are as follows:

1. Apply the potential to every local column of W: $V \cdot w$;
2. For each column of W^T, dispatch a non-blocking gather operation *MPI_Iallgather*;
3. Whenever a gather operation is finished for a local column of W^T, apply B_2 to that column;
4. Compute $W \cdot B_1^T$ after the gather operation has been completed.

In our C++ implementation, the tensor product scheme discussed above is encapsulated as a member function within an operator class. We instantiate the linear operators K and C with this operator class, which contains various attributes, such as differentiation matrices and member functions for operator application. The source code of the function of the parallel operator application is provided below:

```
// y = Op*x = reshape(a1*C*X+a2*X*D', n*m,1)
void apply(const ST* v_in, ST* v_out)
{
    int i,j,k;
    ST ele;

    // V*w
    #pragma omp parallel for
    for (i=0; i<loc_m*n_; i++) v_out[i] = V_[i] * v_in[i];

    // non-blocking allgather operation
    ST* vcol = new ST[m_*n_];
    init(m_, vcol, 0.0);
    MPI_Request request;
    MPI_Iallgather(v_in, loc_m*n_, mpi_type, vcol, loc_m*n_, mpi_type, MPI_COMM_WORLD, &
        request);

    #pragma omp declare reduction(+:ST:omp_out+=omp_in) initializer (omp_priv=omp_orig)
    #pragma omp parallel for reduction(+:ele)
    for (i=0; i<loc_m; i++)
        for (j=0; j<n_; j++)
            for (k=0; k<n_; k++)
            {
                ele =  a2_ * v_in[i*n_+k] * D_[j*n_+k];
                v_out[i*n_+j] += ele;
            }

    MPI_Wait(&request, MPI_STATUS_IGNORE);

    #pragma omp parallel for reduction(+:ele)
    for (i=0; i<loc_m; i++)
        for (j=0; j<n_; j++)
            for (k=0; k<m_; k++)
            {
```

```
                ele = a1_ * vcol[k*n_+j] * C_[(rank*loc_m+i)*m_+k];
                v_out[i*n_+j] += ele;
            }

    delete [] vcol;
    return;
}
```

The provided source codes demonstrate that the original sparse matrix-vector product is decomposed into two dense general matrix-matrix products (*gemm*). However, to achieve better performance and optimize the execution of the *gemm*, we can enhance our current implementation by replacing it with the routines provided by highly optimized mathematical computing libraries. In most cases, our naïve implementation of the matrix-matrix multiplication exhibits slower performance compared to BLAS (Basic Linear Algebra Subprograms), which is a highly optimized mathematical computing library widely used for efficient matrix operations. BLAS leverages specialized algorithms and optimizations to maximize computational efficiency and take advantage of hardware-specific features, resulting in faster matrix computations. Therefore, we choose to replace our trivial nested loop implementation of matrix-matrix products with the more efficient *?gemm* function provided by the BLAS library, where the choice of *?* depends on the data type. For example, *sgemm* is used for single precision (float), *dgemm* for double-precision, and *cgemm* for complex single precision. In our case, when we are dealing with complex double precision, we will utilize the *zgemm* function to ensure accurate and efficient matrix computations. The adaptation of our original implementation is as follows:

```
// y = Op*x = reshape(a1*C*X+a2*X*D', n*m,1)
void apply(const ST* v_in, ST* v_out)
{
    int i,j,k;
    ST ele;

    // V*w
    #pragma omp parallel for
    for (i=0; i<loc_m*n_; i++) v_out[i] = V_[i] * v_in[i];

    // non-blocking allgather operation
    ST* vcol = new ST[m_*n_];
    init(m_, vcol, 0.0);
    MPI_Request request;
    MPI_Iallgather(v_in, loc_m*n_, mpi_type, vcol, loc_m*n_, mpi_type, MPI_COMM_WORLD, &
        request);

    cblas_zgemm(101, 111, 112, loc_m, n_, n_,
                a2_, v_in, loc_m, D_, n_, 1.0,
                v_out, loc_m);
    MPI_Wait(&request, MPI_STATUS_IGNORE);
    cblas_zgemm(101, 111, 111, loc_m, n_, m_,
                a1_, C_+(rank*loc_m)*m_, loc_m, v_in, m_, 1.0,
                v_out, loc_m);

    delete [] vcol;
    return;
}
```

The tensor product scheme introduced in this subsection offers an efficient implementation for the operator application. Beyond its benefits for operator application, the tensor structure can also be exploited for preconditioning. Recalling Subsection 3.2.5, it has been demonstrated that preconditioning entails solving the linear system $P\tilde{\mathbf{w}} = \mathbf{w}$, where P should approximate $T(\theta)$. In the upcoming subsection, we will show that when $P \approx T(\theta)$ also possesses the tensor structure, an efficient algorithm can be used to solve the linear system $P\tilde{\mathbf{w}} = \mathbf{w}$.

4.2.2. Preconditioning for tensor operators

It is also possible to exploit the tensor structure in the implementation of preconditioning. In the last subsection of Chapter 3, we introduce that the application of preconditioner P involves solving the linear system $P\tilde{\mathbf{w}} = \mathbf{w}$. P should be chosen such that it approximately matches $T(\theta)$ and ensures easy

invertibility so that $\tilde{\mathbf{w}} = P^{-1}\mathbf{w}$ can be computed easily. In the 1D three-body system, for some small values of the potential depth v_0, the stiffness matrix K can be approximated by Eq. (4.1) neglecting the potential term $v_0(F_+ + F_-)$:

$$T_{0,a_1,a_2} = a_1(B_1 \otimes \mathbb{I}_2) + a_2(\mathbb{I}_1 \otimes B_2), \tag{4.3}$$

where $B_1 = K_y$ and $B_2 = K_x$. T_{0,a_1,a_2} possesses the same tensor structure with the damping operator $C^{(3b)}$. We are able to preserve this tensor structure while adding stiffness and damping matrices together with the diagonal mass matrix due to the bilinearity and associativity of the Kronecker product. The resulting tensor provides an excellent approximation to $T(\theta)$. However, to lessen the computational burden, we refrain from summing all matrices together to approximate $T(\theta)$ with P. Instead, we only make $P \approx T(\tau)$, where τ is the target value for the JD approximation. The rationale behind this choice will be explained later in this subsection. If τ is set to 0, then T_{0,a_1,a_2} given by Eq. (4.3) corresponds exactly to our preconditioner.

Once we have a preconditioner P in the formula of Eq. (4.3), the next step is to solve the linear system with T_{0,a_1,a_2} and some right-hand side $\mathbf{w}$. To improve the stability, some shift σ can be introduced. For some scalar σ, the linear system $(T_{0,a_1,a_2} - \sigma\mathbb{I})\tilde{\mathbf{w}} = \mathbf{w}$ can be solved through the Sylvester equation

$$(B_1 - \sigma_1\mathbb{I})W + W(B_2 - \sigma_2\mathbb{I})^T = B, \tag{4.4}$$

where $B = \text{reshape}(b, N_y, N_x)$ and $\sigma = \sigma_1 + \sigma_2$. The investigation of an optimal value for σ is beyond the scope of our research study. To solve Eq. (4.4), we can employ the Bartels–Stewart algorithm (Alg. 5) [4], a direct method for solving the Sylvester equation. It requires a Schur decomposition of the shifted matrices $B_{1,2} - \sigma_{1,2}\mathbb{I}$, a combination of two dense matrix-matrix products, and a forward/backward substitution with the Schur factors.

Algorithm 5 The Bartels–Stewart algorithm computing $AX + XB = C$

Input: $A \in \mathbb{R}^{m\times m}$, $B \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{m\times n}$
Output: $X \in \mathbb{R}^{m\times n}$
Compute the real Schur decompositions: $R = U^T AU$, $S = V^T B^T V$
Set $F = U^T CV$
Solve the system $RY - YS^T = F$, where $T = U^T XV$ using forward substitution on the blocks.
Set $X = UYV^T$

Fortunately, we do not need to implement Alg. 5 by ourselves. In MATLAB, we can simply utilize the built-in command *sylvester* to solve Eq. 4.4, which greatly simplifies the process and saves us time and effort. In our C++ implementation, we take advantage of LAPACK (Linear Algebra Package) to efficiently solve the Sylvester equation. LAPACK is a widely-used mathematical computing library that provides a comprehensive set of routines for solving linear algebra problems. It is designed to efficiently handle various numerical computations involving matrices, vectors, and linear equations. LAPACK provides the *?trsyl* routine (the *?* represents either *s* for real single-precision, *d* for real double-precision, *c* for complex single-precision, or *z* for complex double-precision) to solve the Sylvester equation for real quasi-triangular or complex triangular matrices.

```
// approximate inverse operation, y = Op\b
void invapply(ST* b, ST* v_out)
{
    lapack_int info;
    double scale=1.0;
    double* pscale = &scale;
    vec_update(n_*m_, scale, b, v_out);
    info = LAPACKE_ztrsyl(101, 'N', 'C', 1, m_, n_, C_, m_, D_, n_, v_out, m_, pscale);
}
```

ztrsyl requires the triangular matrices as inputs. We employ Schur decomposition to obtain the Schur forms of $B_{1,2} - \sigma_{1,2}\mathbb{I}$. These Schur forms are then passed to *ztrsyl*. Now we can explain we only make the preconditioner P approximate $T(\tau)$ instead of $T(\theta)$. Given that $T(\theta)$ depends on the Ritz value θ, which varies with each JD iteration, it is necessary to compute the Schur decomposition in

every iteration. However, by fixing the value at τ, the decomposition only needs to be computed once, leading to significant computational savings.

4.3. Some remarks

It is worth noting that the parallel Jacobi-Davidson algorithm we have implemented exhibits a degree of parallelism, but it is not entirely parallel throughout the entire iteration process. According to Algorithm 4, throughout the entire process of the Jacobi-Davidson iteration, each compute core is tasked with solving the projected eigenvalue problem. Consequently, the linearization operation, as well as the utilization of mathematical computing libraries to solve the linearized problem, are performed sequentially by all compute cores. However, the impact of this sequential computation is mitigated due to the relatively small size of the projected eigenvalue problem, which typically remains below the maximum restart dimension. Consequently, the computational overhead incurred by the sequential computation of solving projected QEPs is relatively modest.

Another significant computational aspect worth noting arises from the process of preconditioning. As introduced in Subsection 4.2.2, the preconditioning step involves solving the Sylvester equation by calling the LAPACK routine *?trsyl*. In contrast to the projected QEP-solving process, where the problem size remains quite small, the preconditioning step involves much more computations. The need for each compute core to call the LAPACK routine *?trsyl* can indeed result in considerable computational overhead, particularly because preconditioning is performed at each application of the linear operator during the GMRES iterations. The application of the operator with preconditioning takes significantly more time than the one without preconditioning. Therefore, one may question whether it is worthwhile to perform preconditioning, considering the great amount of sequential computation involved. We are now confronted with a trade-off between convergence speed and computational speed. It is noteworthy that the reduction of preconditioning's computational time is attainable through the utilization of OpenMP parallelism in LAPACK routines on shared memory machines. Regrettably, we have not harnessed the parallelism of *?trsyl* until the completion of this thesis. Nevertheless, in Chapter 5, we will show that sequential preconditioning remains advantageous as it helps reduce computational time by expediting convergence significantly.

The numerical experiments of the parallel Jacobi-Davidson algorithm were performed on the DelftBlue supercomputer at TU Delft [1]. Each compute node of DelftBlue consists of two Intel-Xeon E5-6248R processors with 24 cores and has 192 GB of RAM. Later in the second section of Chapter 5, we will demonstrate that in our 1D three-body problem, we utilize up to 256 grid points individually along both the x and y coordinates. That is to say, the size of each vector is 256$\times$256, requiring exactly

$$\frac{256 \times 256 \times 16 \text{ Bytes (complex double)}}{1024 \times 1024} = 1 \text{ MB}$$

of memory. It is obvious that we can fit thousands of vectors on a single compute node of DelftBlue. Similarly, with respect to the efficiency of *gemm* operations, the operation is significantly far from being compute-bound on a single compute node. As a consequence, attempting to scale beyond one compute node would yield limited benefits, despite the fact that the algorithm supports both shared and distributed memory architectures.

5

Numerical Results

The chapter on numerical results serves as a crucial part, presenting all outcomes derived from our algorithms and addressing the research questions. We will showcase the numerical results obtained from computing both quantum two-body and three-body problems, respectively. When addressing the computation of two-body systems, we first introduce a special case, the Pöschl-Teller potential, as a benchmark to meticulously verify the correctness of the numerical algorithm. Subsequently, we will make use of a two-body test problem to evaluate the performance of our Jacobi-Davidson eigensolver. After addressing the two-body problems, our focus shifts to the section dedicated to one-dimensional three-body problems. The parallel Jacobi-Davidson algorithm introduced in Chapter 4 is utilized for computing the resonant states of three-body problems.

5.1. Two-body problems

We commence by investigating quantum two-body problems, which are easier to address compared with three-body systems. We formulate the quadratic eigenvalue equations following the process introduced in Subsection 2.2.1. In Subsection 5.1.1, a two-body system featuring a special potential called the Pöschl-Teller potential is introduced. This potential possesses analytical solutions for bound states and resonant states [9], allowing us to compare them with our numerical results. This comparison serves as a validation step to ensure the correctness of our numerical method. After verifying the correctness, we extend our analysis to two-body systems with arbitrary Gaussian potentials. We will discuss the research questions regarding the performance of eigensolvers in scenarios involving Gaussian potentials in Subsection 5.1.2 and 5.1.3.

5.1.1. A nice benchmark: Pöschl-Teller potential

Typically, solving the stationary Schrödinger equation for two-body systems analytically is a challenging task. However, the Pöschl-Teller potential stands as an exceptional case. Cevik et al. [9] demonstrates that the Hamiltonian featuring the Pöschl-Teller potential exhibits a remarkable property: all the poles of the S matrix associated with the purely outgoing boundary conditions, which reveal the energies and momenta of bound, antibound, and resonance states, can be derived analytically and with absolute precision. This is a very exceptional outcome since these poles can only be obtained by numerical methods for the vast majority of worked potentials. As a result, the Pöschl-Teller potential can serve as an ideal benchmark against which we can validate the accuracy of our numerical results.

The real hyperbolic Pöschl-Teller potential is given by

$$V_{pt} = -\frac{\hbar^2}{2m}\frac{\alpha^2\lambda(\lambda-1)}{\cosh^2\alpha x}, \tag{5.1}$$

where α is a fixed constant while λ is a parameter. In the following analysis, we neglect the term $\hbar^2/2m$ and focus solely on the dimensionless form of (5.1). There are three possibilities for λ each one giving a different shape for the potential:

- $\lambda > 1$, potential well,
- $\frac{1}{2} \leq \lambda < 1$, low barrier,
- $\lambda = \frac{1}{2} + il; l > 0$, high barrier.

The justification for the assigned names comes from their shapes shown in Fig. 5.1. For the value $\lambda = 1$ the potential vanishes. For integer values of λ, greater than one, it is well known that the resulting potential is reflectionless.

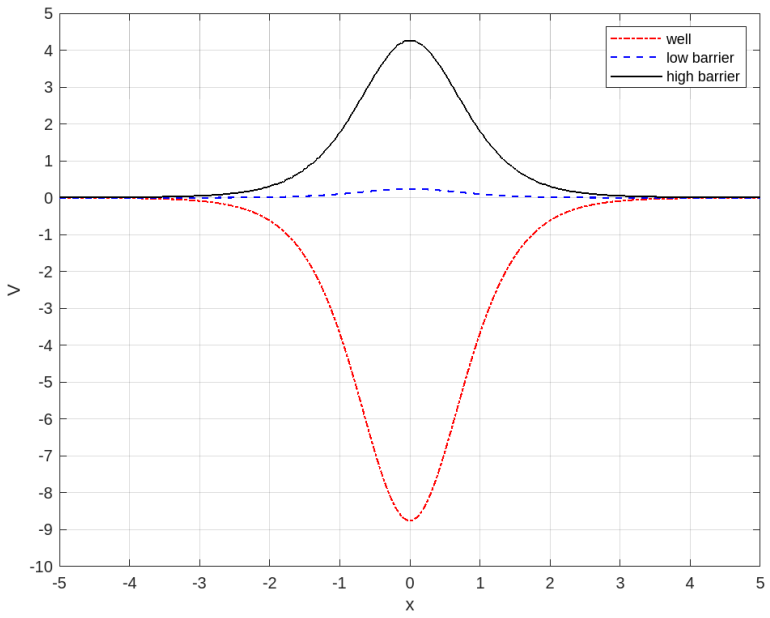
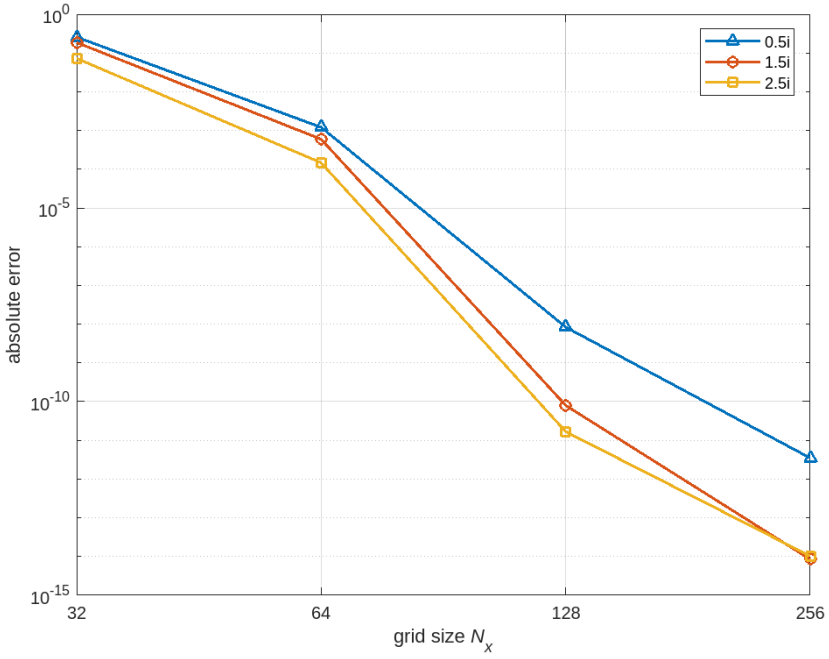
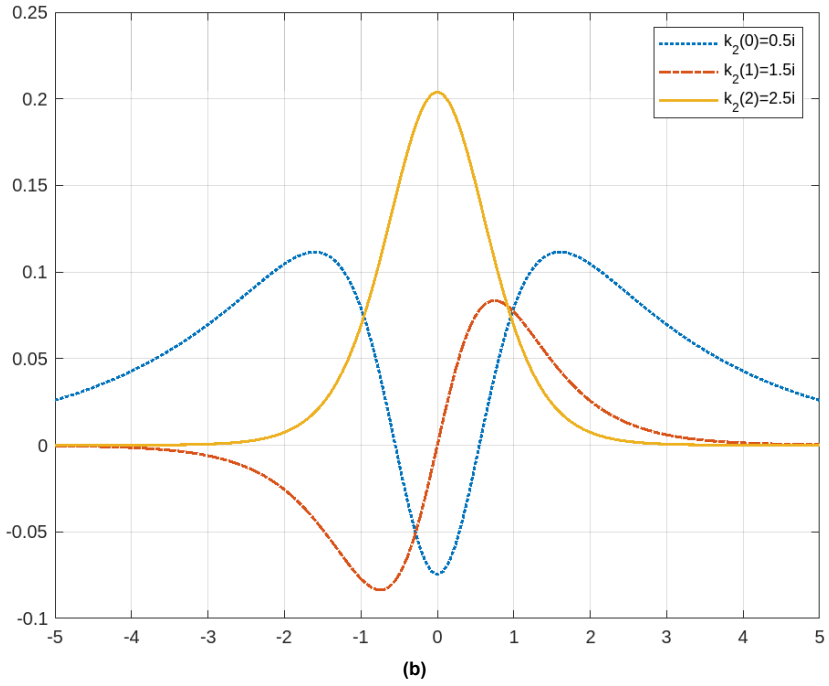


Figure 5.1: Plot of the Pöschl-Teller potential for different values of λ: $\lambda = 3.5$ (well), $\lambda = 0.6$ (low barrier), $\lambda = 0.5 + 2i$ (high barrier).

The property of Hamiltonian with different λ has been well studied by [9]. It is shown that when $\lambda > 1$, where λ is not an integer, there exists an infinite number of antibound poles, all of which are situated in the negative region of the imaginary axis. A few bound states can be found in the positive portion of the axis. When $\lambda = 3.5$, there are three bound states on the positive imaginary axis: $0.5i$, $1.5i$, and $2.5i$. Fig. 5.2a shows that the numerical solutions of our discretization can successfully converge to the bound states. And the wave functions corresponding to the bound states are illustrated in Fig. 5.2b.



(a)



(b)

Figure 5.2: (a) Absolute error between the numerical results and analytical solutions for three bound poles when $\lambda = 3.5$. (b) Wave functions of three bound states.

It is shown in [9] that for the case of a low barrier, where $\frac{1}{2} \leq \lambda < 1$, no bound states exist; instead, anti-bound states can be detected. For $\lambda = 0.75$, the series of the anti-bound states is given by $-0.25i$, $-1.25i$, $-2.25i$, $\cdots$. Fig. 5.3a demonstrates the convergence of the numerical solution to the anti-bound state $-0.25i$. On the other hand, the resonance poles only appear in the high barrier, when $\lambda = \frac{1}{2} + il$. The S matrix has an infinite number of resonance poles that appear in pairs symmetrically located with respect to the negative part of the imaginary axis. Assuming that $l > 0$, both series of pole solutions can be written as:

$$k_1(n) = l - i(n + \frac{1}{2}), \quad k_2(n) = -l - i(n + \frac{1}{2}), \tag{5.2}$$

where $n = 0, 1, 2, \cdots$. For each value of n, solutions $k_1(n)$ and $k_2(n)$ give a pair of resonance poles. Fig. 5.3b illustrates the convergence of the resonance poles towards $\pm 2 - 0.5i$ $(k_{1,2}(0))$ as the number of grid points increases, given $\lambda = \frac{1}{2} + 2i$.

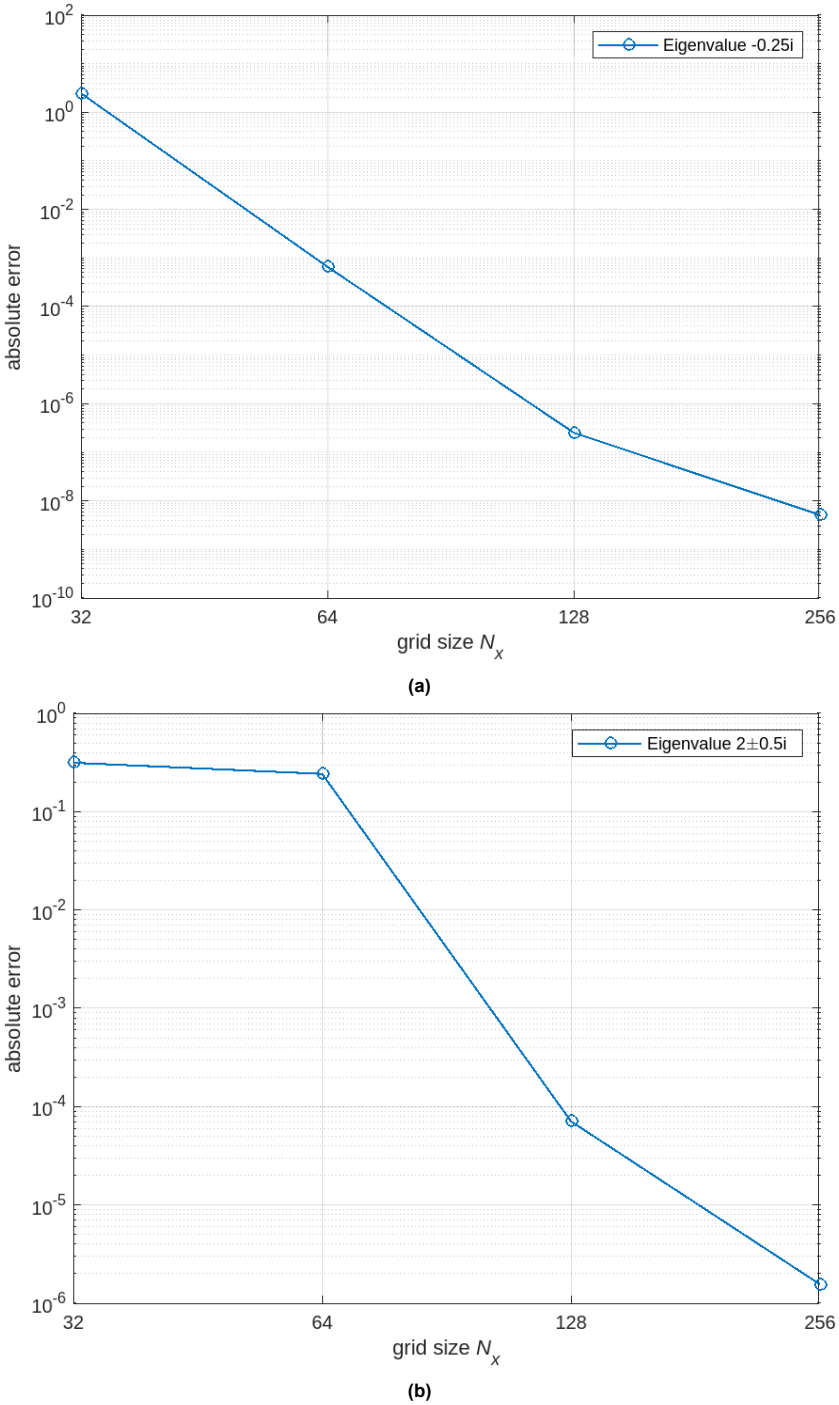


Figure 5.3: (a) Absolute error between the numerical results and analytical solutions for the anti-bound pole $-0.25i$ when $\lambda = 0.75$. (b) Absolute error between the numerical results and analytical solutions for the resonance poles $2 \pm 0.5i$ when $\lambda = \frac{1}{2} + 2i$.

Thus far, we can justify the capability of our numerical algorithms to identify resonance poles and bound poles, as evidenced by the overlap between the analytic and numerical solutions of the Pöschl-Teller two-body system. However, as indicated in Chapter 1, it is common for the majority of few-body potentials to decay either exponentially (Gaussian shape) or polynomially (Lorentzian shape) as $x \to \infty$.

The Pöschl-Teller potential, along with the square potential discussed in the beginning, is atypical and uncommon for few-body potentials. As a result, in the following subsections, we will substitute the potential with a Gaussian shape. To evaluate the performance of eigensolvers, we will utilize a two-body problem featuring a Gaussian potential as a test case.

5.1.2. Answer to the research question: Comparison of eigensolvers

Now we extend our analysis from the investigation of the Pöschl-Teller potential to a more practical two-body potential. We now consider two quantum particles bound by Gaussian potential given by

$$V(x) = -v_0 \exp(-x^2). \tag{5.3}$$

v_0 represents the depth of the potential, the determination of which is beyond the scope of this article. To test our eigensolvers, we simply make use of the value used by [27], $v_0 = 0.34459535$ corresponding to the two-body binding energy $\mathcal{E}_0^{(2)} = 10^{-1}$. In this two-body problem, we keep the implementation in MATLAB. We have developed MATLAB implementations for two eigensolvers: linearization combined with the Jacobi-Davidson QZ algorithm, and the nonlinear (quadratic) Jacobi-Davidson algorithm. To compare the convergence speed and scalability of the two solvers, we apply both solvers to increasingly larger problem sizes, with the maximum size reaching 2048. The performance comparison within 50 iterations is illustrated by Fig. 5.4 and 5.5. In addition to comparing JDQZ and JD, we also analyze the performance difference between using preconditioning and not using preconditioning for separate solvers. The preconditioner, which is analogous to the shifted Hamiltonian without potential employed in [27], is the stiffness matrix K with some shifts ignoring the potential component. Preconditioning is necessary, as shown in Fig. 5.4 and 5.5, particularly for JDQZ. Without the application of preconditioning, JDQZ fails to converge entirely during the initial 50 iterations. Similarly, although the quadratic Jacobi-Davidson algorithm may exhibit very slow convergence, it also fails to decrease the residual norm when dealing with large problem sizes.

Some remarks regarding the settings of JDQZ and JD solvers are as follows: The tolerance to the residual norm $\|r\|_2$ is set to 10^{-8}. Considering that the correction equation does not require to be solved very accurately, the maximum number of iterations for the GMRES solver is set to 30. To mitigate an excessively large search space, we impose a restart restriction on the subspace dimension, limiting it to a range between 10 and 30. The target for the approximation is set to $0 + 1i$, the selection of which is determined based on the pre-information obtained from the coarse grid. We employ MATLAB's **eig** command to solve a small quadratic eigenvalue problem obtained from the coarse grid. This allows us to gain a first impression of the distribution of resonance poles. The position $0 + 1i$ is considered an optimal location where resonance poles concentrate.

Problem size	Number of eigenvalues detected	
	prec-JD	prec-JDQZ
128	9	18
256	13	7
512	10	7
1024	9	6
2048	0	1

Table 5.1: Number of eigenvalues detected by 50 iterations of preconditioned JD and JDQZ.

Table 5.1 shows a comparison of the number of eigenvalues found below the tolerance by 50 iterations of two solvers. We observe that for relatively smaller problem sizes, such as 128 grid points, the JDQZ algorithm proves to be a favorable choice. However, as the problem size increases, on average, the Jacobi-Davidson method is able to discover more eigenvalues within 50 iterations. Both solvers require more than 50 iterations to converge when the number of grid points reaches 2048. When comparing the convergence rates of JD and JDQZ, determining the superior performer proves challenging. Thus, we can only conclude that preconditioning remains essential for both quadratic JD and JDQZ in terms of convergence speed. Nevertheless, the disparity in convergence speed between quadratic JD and JDQZ diminishes when addressing large problem sizes. To further evaluate their performance, we measure the computational time of both methods to assess if there is a significant difference.

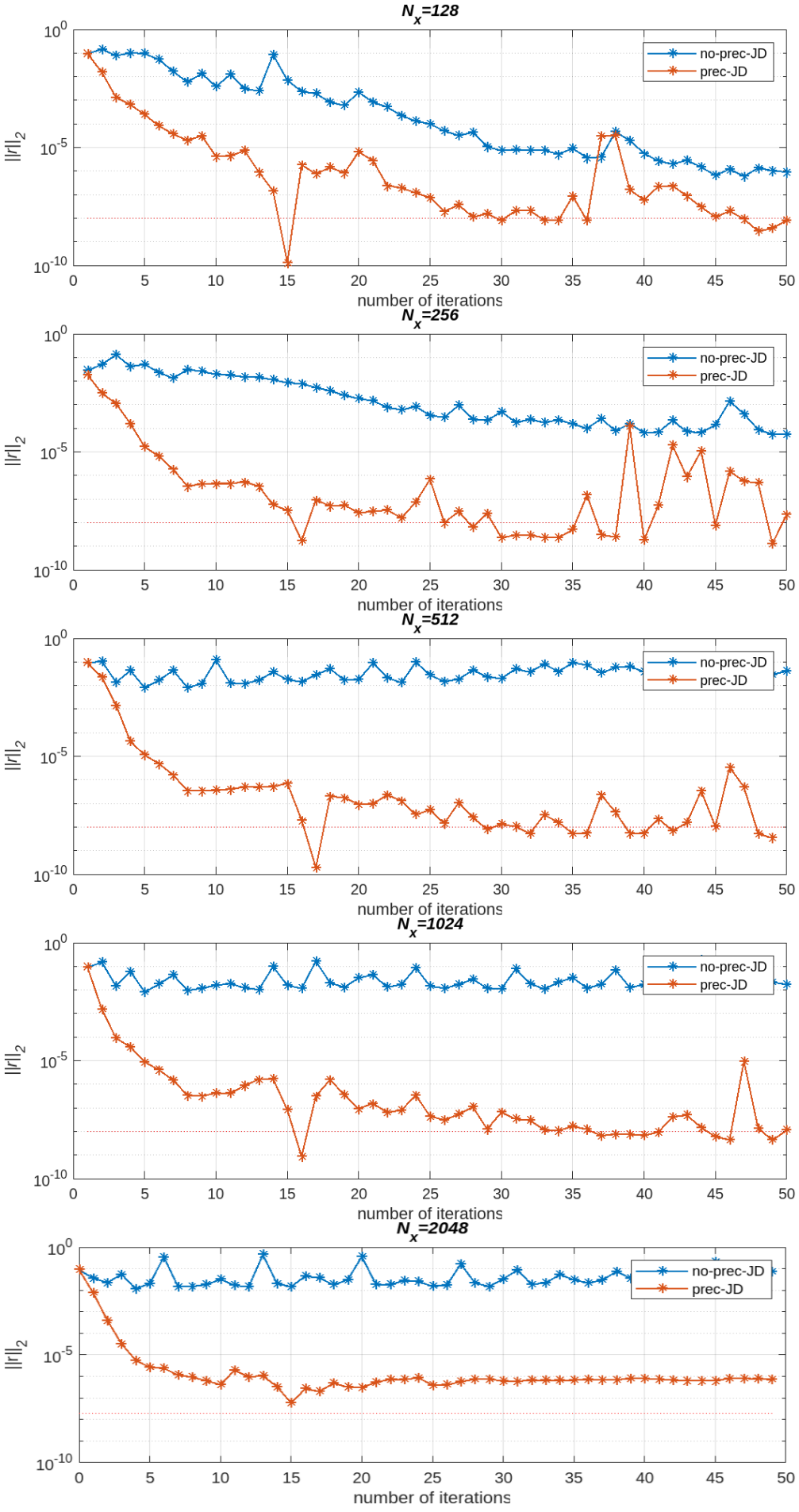


Figure 5.4: Convergence history of the quadratic Jacobi-Davidson method with and without preconditioning varying as the problem size increases.

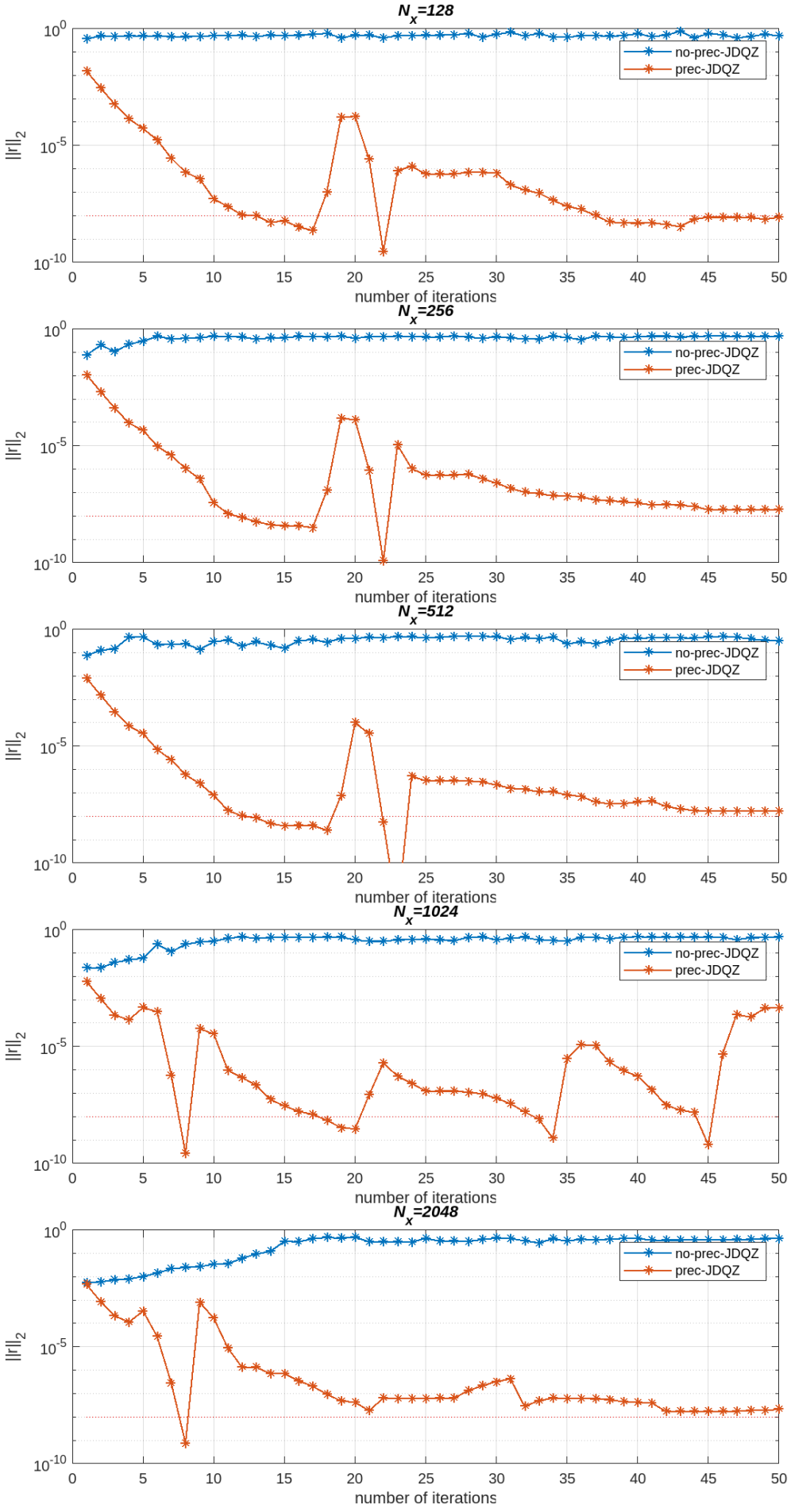


Figure 5.5: Convergence history of the Jacobi-Davidson QZ method with and without preconditioning varying as the problem size increases.

Figure 5.6 illustrates the noticeable gap in computational time between preconditioned JD and JDQZ. It is evident that JDQZ requires significantly more time to complete 50 iterations. The result is not surprising since the generalized eigenvalue problem solved by JDQZ is twice the size of the problem handled by JD, due to linearization. Moreover, JDQZ entails more operations, resulting in a longer running time for a single iteration in comparison to JD, even when the problem sizes are identical. Therefore, we can infer that the combination of linearization and the Jacobi-Davidson QZ method is not efficient for solving the large-scale quadratic eigenvalue problem when contrasted with the quadratic Jacobi-Davidson method. Note that our current focus has been on two-body problems. Another fatal drawback of linearization is the infeasibility of leveraging the efficient tensor product of the three-body problems. This is because the process of linearization reduces the order of the original problem, destroying the tensor structure present in the stiffness and damping operators. As a result, when tackling three-body problems, the preferred approach is to directly apply the Jacobi-Davidson method to quadratic eigenvalue problems. As discussed in Chapter 4, our research solely focuses on utilizing the Jacobi-Davidson method and investigating its parallelism for three-body problems.

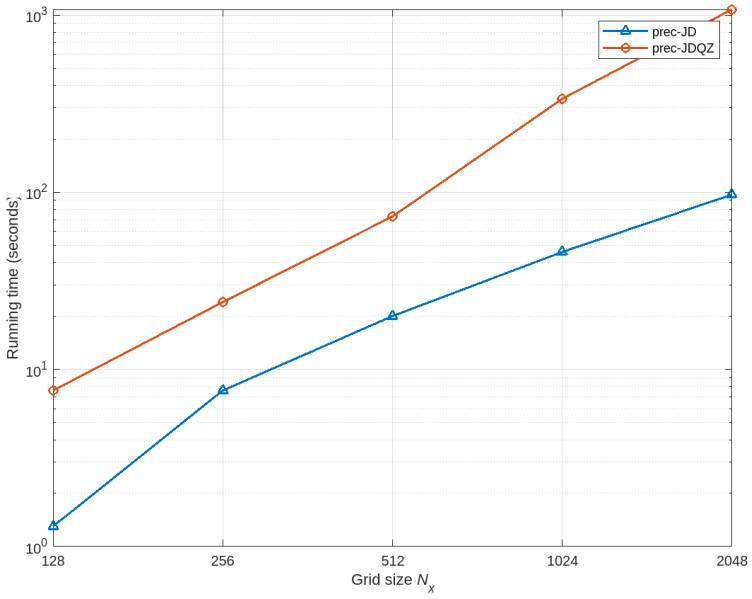


Figure 5.6: The running times of JD and JDQZ computing 10 eigenvalues as a function of the grid size N_x.

In conclusion, the response to research question 2 is as follows: The convergence speeds of the two eigensolvers, quadratic JD and JDQZ do not differ so much, making it challenging to definitively determine the superior method. However, the difference in computational time between the solvers suggests that the quadratic Jacobi-Davidson algorithm is a preferable choice, particularly for addressing three-body problems. Furthermore, this subsection provides a partial answer to the research question regarding preconditioning. Preconditioning proves essential for both solvers in accelerating the convergence process. A more detailed examination of preconditioning in the context of three-body problems will be presented in Section 5.2. The subsequent subsection will address research question 3: How can the repeated detection of eigenvalues in the quadratic Jacobi-Davidson algorithm be mitigated?

5.1.3. Answer to the research question: Preventing repetition of eigenpairs

In Chapter 3, we discussed the potential risk of converging to the same eigenvalue multiple times during the Jacobi-Davidson iterations. Several works aimed at solving this problem were referenced in that chapter. In our JD algorithm, we have opted to employ a technique known as ”locking”, i.e. putting already converged eigenvectors into the search space.

We compare the count of distinct eigenvalues found by a specific number of Jacobi-Davidson iterations with both ”locking” and ”non-locking” techniques. The results of this comparison are outlined in Tab. 5.2. Based on the table, it is evident that ”locking” proves beneficial in preventing repeated convergence. As the problem size increases, all eigenvalues identified by ”locking” JD are distinct. Fur-

thermore, it's noteworthy that "locking" also influences the total number of eigenvalues discovered. For instance, when dealing with a problem size of 256, the "locking" JD discovers 6 eigenvalues, whereas the "non-locking" JD finds 5, indicating 1 fewer eigenvalue compared to the "locking" JD with the same number of iterations.

Problem size	(Number of distinct eigenvalues) / (Number of eigenvalues detected)	
	locking	non-locking
128	7/10	5/10
256	6/6	4/5
512	4/4	2/4

Table 5.2: Number of distinct eigenvalues detected by JD with "locking" and "non-locking" for various problem sizes.

The "locking" technique has demonstrated its efficacy in preventing the convergence to the same eigenvalue multiple times during JD iterations. In addition to "locking", we have also attempted to filter out repeated eigenvalues using a specific selection criterion proposed by [17]. However, regrettably, our implementation of this selection criterion did not lead to an improvement in the ratio of the number of distinct eigenvalues to the total number of eigenvalues detected. For detailed information regarding the implementation of the selection criterion, please refer to Appendix A.

5.1.4. Resonant states of arbitrary Gaussian potentials

While assessing the performance of the eigensolvers on the test problems, we concurrently obtain the outcomes of converged eigenvalues. These eigenvalues hold physical interpretations, representing resonance poles in the context of our study. Now we employ a sequence of grid sizes N_x to assess the convergence of the eigenvalues. Figure 5.7 illustrates the convergence of the relative spatial discretization error

$$\delta\mathcal{E} = \frac{|\mathcal{E}(N_{x_1}) - \mathcal{E}(2N_{x_1})|}{|\mathcal{E}(N_{x_1})|} \tag{5.4}$$

for four eigenvalues. The relative difference of the computed eigenvalues on successive grids continues to decrease until it reaches a threshold of 10^{-8}. Figure 5.7 displays two bound poles ($0.4470i$, $-0.9402i$) and two pairs of resonance poles ($\pm 1.0899 - 1.6329i$, $\pm 1.6311 - 2.0835i$) of a two-body system ($v_0 = 0.34459535$). It is worth noting that the positions of these states depend on the potential depth v_0. As shown in Fig. 5.8, the eigenvalues exhibit an upward shift as the potential depth v_0 transitions from 0 to 1. This observation highlights the sensitivity of the pole positions to variations in v_0.

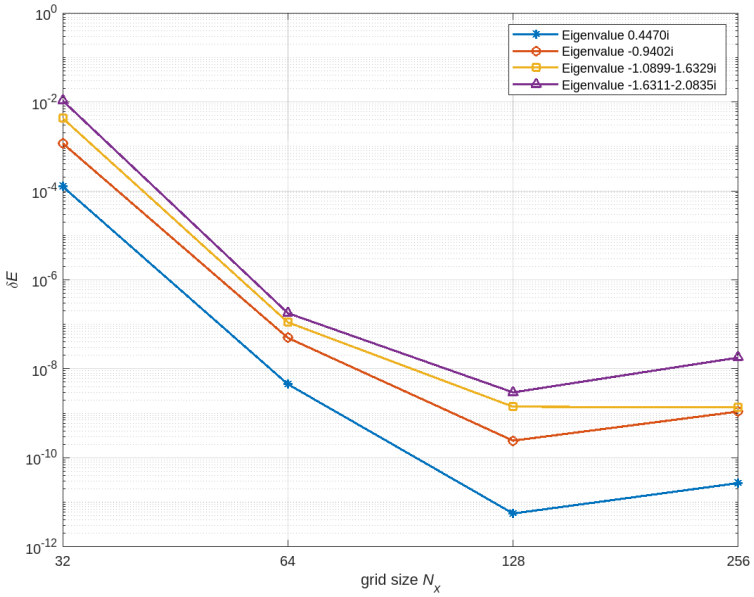


Figure 5.7: Relative spectral discretization error δE as the function of the grid size N_x.

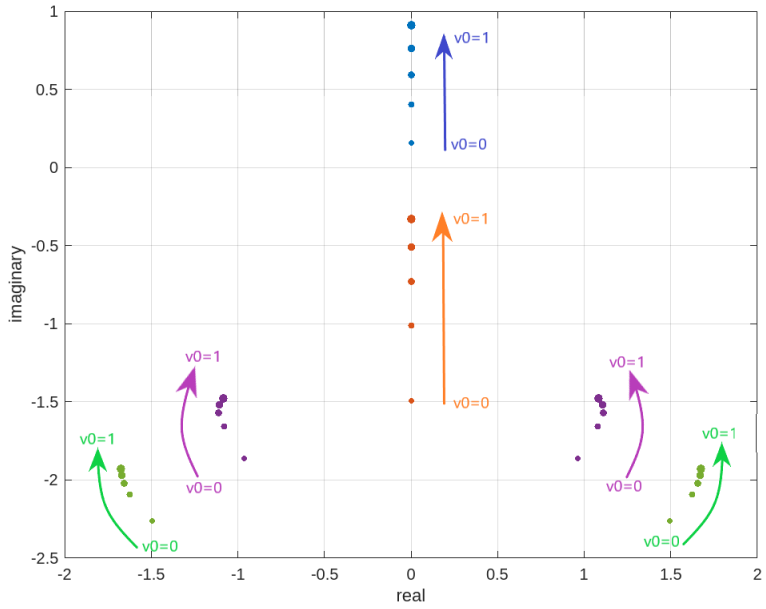


Figure 5.8: Motion of the states(eigenvalues) depending on v_0.

5.2. Three-body problems in 1D

Now we shift our focus towards 1D three-body problems. We consider a test problem involving two non-interacting heavy particles and a light particle with a mass ratio of $\alpha = M/m = 20$, where the potential exhibits a Gaussian shape. As discussed in the preceding sections, the computational resource required for solving three-body problems experiences a rocket-up as a consequence of the increased spatial dimensions involved. Therefore, we need to exploit parallel computing introduced in Chapter 4 to speed up the Jacobi-Davidson algorithm. This section aims to show the results of computing the 1D three-body problem on supercomputers and the performance of the parallel algorithm. We will address research questions 4 and 5 separately in Subsection 5.2.1 and 5.2.2. Finally, we will present the converged resonance poles of the 1D three-body test problems.

5.2.1. Answer to the research question: Preconditioning

For a coarse grid, it is usually easy for the Jacobi-Davidson algorithm to converge to several eigenvalues. However, when dealing with a much finer grid, such as $(N_x, N_y) = (128, 128)$ or $(256, 256)$, the Jacobi-Davidson algorithm may require hundreds of iterations to converge to one eigenvalue, especially when the preconditioner is absent. Fig. 5.9 shows the convergence history of the Jacobi-Davidson method on a $(128, 128)$ grid. The algorithm takes 400 iterations to successfully detect two eigenvalues. Therefore, in order to enhance the convergence of JD, we employ the approach outlined in Subsection 4.2.2, leveraging the Bartels-Stewart algorithm for preconditioning the correction equation within JD. The preconditioner P is initially set to approximate $K^{(3b)} + \sigma C^{(3b)} + \sigma^2 M^{(3b)}$, given by

$$P = (K_x + \sigma C_x) \otimes \mathbb{I}_y + \mathbb{I}_x \otimes (K_y + \sigma C_y), \tag{5.5}$$

where K_x and C_x are shifted slightly. Due to the lack of tensor structures in the mass operator M and potential V, including both of them in our approximation can indeed be problematic and potentially introduce complications. However, in practice, the approximation still proves to be effective and yields satisfactory results. By fixing the approximation at the target value σ, we eliminate the need to compute the Schur decomposition, which is required by the Bartels-Stewart algorithm, for varying matrices in each JD iteration. This significant reduction in computational time leads to substantial time savings. However, the effectiveness of $P \approx K^{(3b)} + \sigma C^{(3b)} + \sigma^2 M^{(3b)}$ diminishes as the number of iterations increases. This fixed preconditioner fails to yield much acceleration for GMRES once a certain number of eigenvalues have been converged. Therefore, when such a case arises, it becomes necessary to replace σ in Eq. (5.5) with θ, representing the Ritz value in the current JD iteration. The new preconditioner should be utilized for a certain number of iterations until it is no longer effective. By employing this approach, we save computational time by avoiding the computation of the Schur decomposition for

every operator application, while consistently achieving an effective preconditioner that ensures fast convergence.

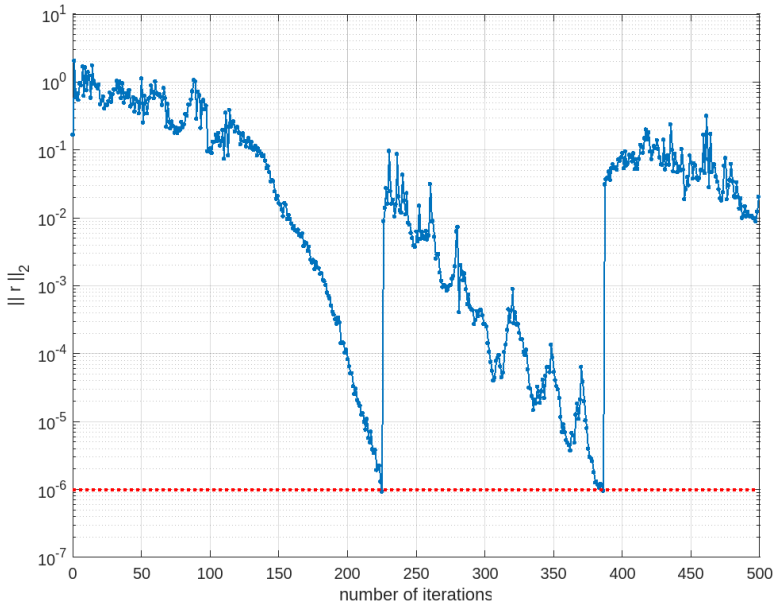


Figure 5.9: Convergence history of the Jacobi-Davidson algorithm on a $(128, 128)$ grid.

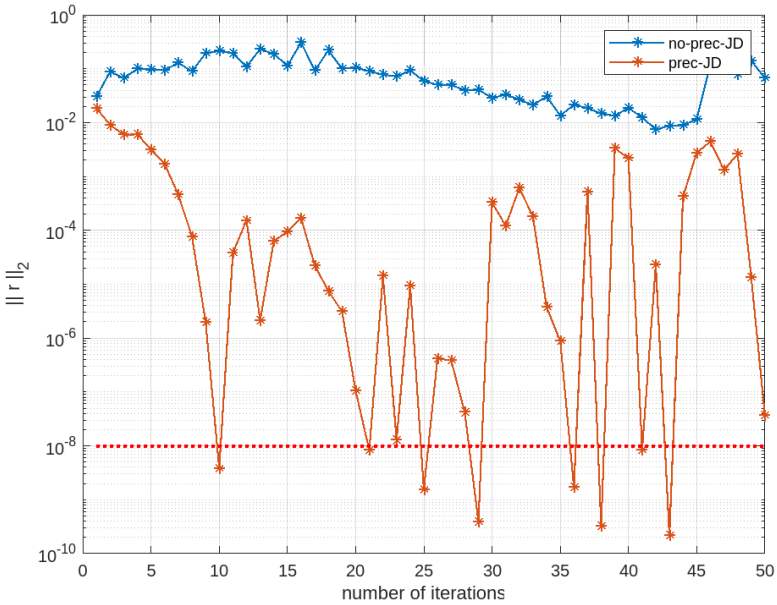


Figure 5.10: Convergence history of 50 Jacobi-Davidson iterations with and without preconditioning on a $(128, 128)$ grid.

In Chapter 4, we discussed the feasibility of harnessing shared-memory parallelism while utilizing LAPACK for applying the preconditioner in our parallel JD implementation. However, regrettably, our current implementation of the preconditioner application remains sequential and computationally intensive. As a result, we appear to be faced with a trade-off between achieving shorter computational times per JD iteration and achieving faster overall convergence. However, our practical experience has demonstrated that despite the preconditioner increasing the running time of a single iteration, it ultimately saves computational time by significantly reducing the number of iterations required to converge. Fig. 5.10 illustrates the significant improvement in the convergence of JD achieved through preconditioning. In contrast to the hundreds of iterations required for the non-preconditioned Jacobi-Davidson

method to converge to a single eigenvalue, the application of preconditioning enables convergence to several eigenvalues in just 50 iterations. In conclusion, the preconditioner inspired by the approach outlined in [27] is highly effective in increasing the convergence speed and reducing the required computational time. This result addresses research question 4.

5.2.2. Answer to the research question: Performance of parallelization

The numerical experiments of the parallel Jacobi-Davidson algorithm were performed on the DelftBlue supercomputer at TU Delft [1] with up to 64 CPU cores. Each compute node of DelftBlue consists of 48 cores, so the program is at most run on two compute nodes with separate memory. In order to investigate the convergence behavior of eigenvalues, we run the program separately on a grid of different dimensions, including $(N_x, N_y) = (32, 32)$, $(64, 64)$, $(128, 128)$, and $(256, 256)$. Figure 5.11 displays how the running time of 50 parallel Jacobi-Davidson iterations depends on the number of compute cores for various problem sizes.

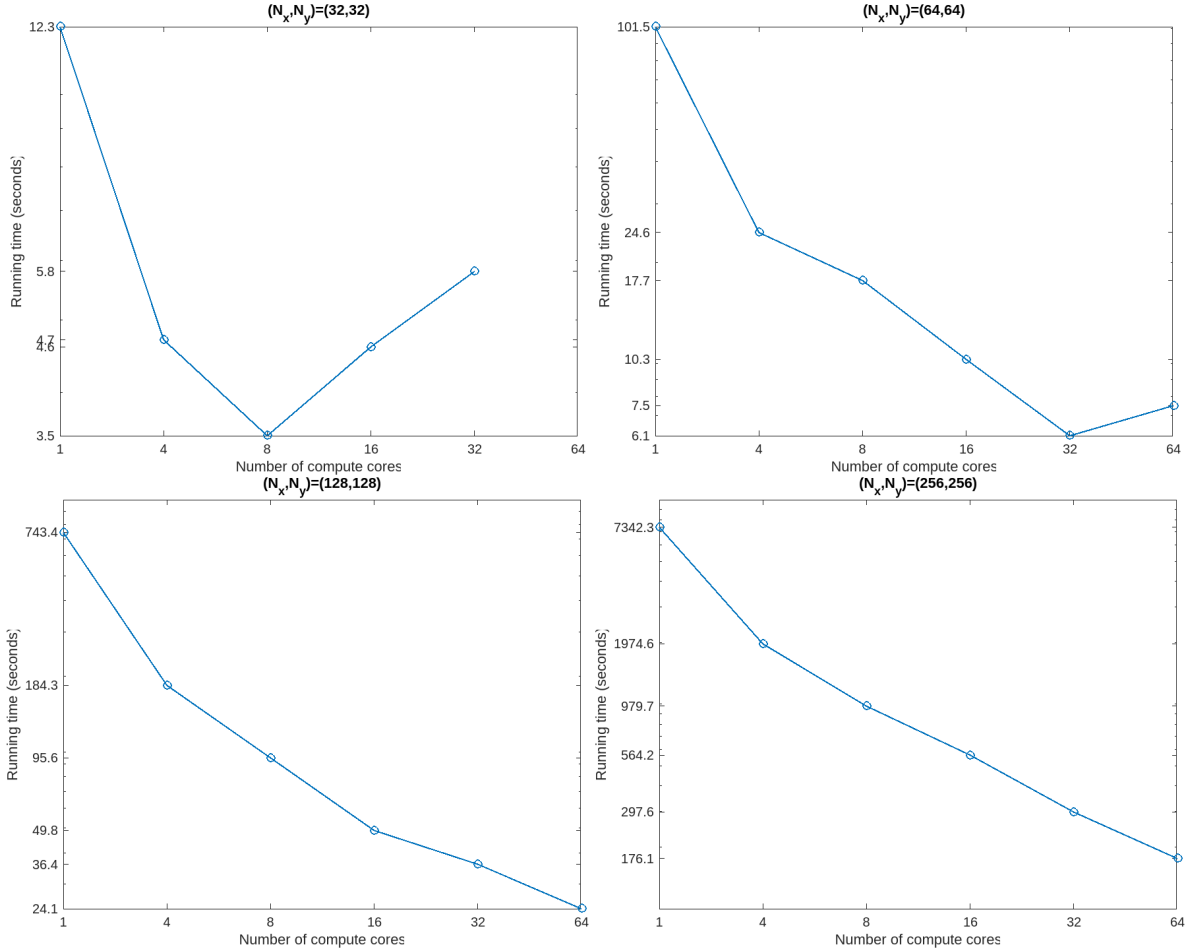


Figure 5.11: Running time of 50 parallel Jacobi-Davidson iterations as a function of the number of compute cores for various problem sizes.

Figure 5.11 proves a statement discussed in Chapter 4: The utilization of parallel computing does not always yield advantages in terms of reducing computational time. When dealing with small problem sizes, such as $(N_x, N_y) = (32, 32)$ or $(64, 64)$, the problem becomes highly memory-bound. The overhead resulting from excessive memory traffic due to an abundance of compute cores outweighs the benefits of reduced computational complexity per individual compute core. Consequently, as shown in Fig. 5.11, the running time increases when an excessive number of compute cores are utilized for $(N_x, N_y) = (32, 32)$ or $(64, 64)$.

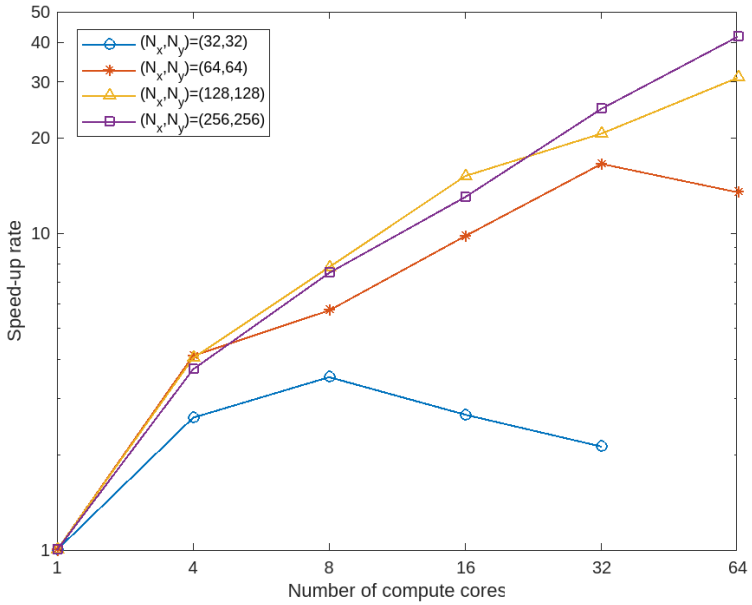


Figure 5.12: Speedup rate of the parallel Jacobi-Davidson algorithm for various problem sizes.

In the chapter on parallelization, we argued that the small size of a single vector allows us to fit thousands of vectors on a single compute node of DelftBlue. As a consequence, the operation is significantly far from being compute-bound and using more compute nodes yields limited performance increases or even slower speeds. To provide a more direct illustration, we employ the concept of speedup. Speedup in high-performance computing can be defined as the ratio of the time taken by serial execution and the time taken for parallel execution. Figure 5.12 illustrates a decline in the speedup rate when an excessive number of compute cores are employed for $(N_x, N_y) = (32, 32)$ or $(64, 64)$. Additionally, we can anticipate that for $(N_x, N_y) = (128, 128)$ or $(256, 256)$, the speedup improvement will gradually diminish as the number of compute cores surpasses 64.

5.2.3. Convergence of resonant states

Drawing inspiration from our previous endeavors in two-body problems, we utilize a series of grid sizes (N_x, N_y) to evaluate the convergence behavior of the eigenvalues. We observe that the relative discretization error of three eigenvalues converges to a level within the tolerance of the eigensolver, set at 10^{-6}. This convergence implies that as the grid size increases, the numerical approximation of the eigenvalues approaches the true values with a high degree of accuracy, satisfying the desired precision criterion defined by the eigensolver's tolerance.

Fig. 5.13 illustrates the convergence of three eigenvalues, which can be identified as the resonance poles of the 1D three-body system. It's important to note that the 1D three-body system we are addressing is not associated with any real physical model. Given that the primary focus of this thesis project is the mathematical solver, an in-depth exploration of the physics underlying the three-body resonant model has not yet been undertaken. Nonetheless, the convergence depicted in Fig. 5.13 highlights the efficacy of our parallel Jacobi-Davidson algorithm in detecting resonant states within arbitrary 1D three-body systems.

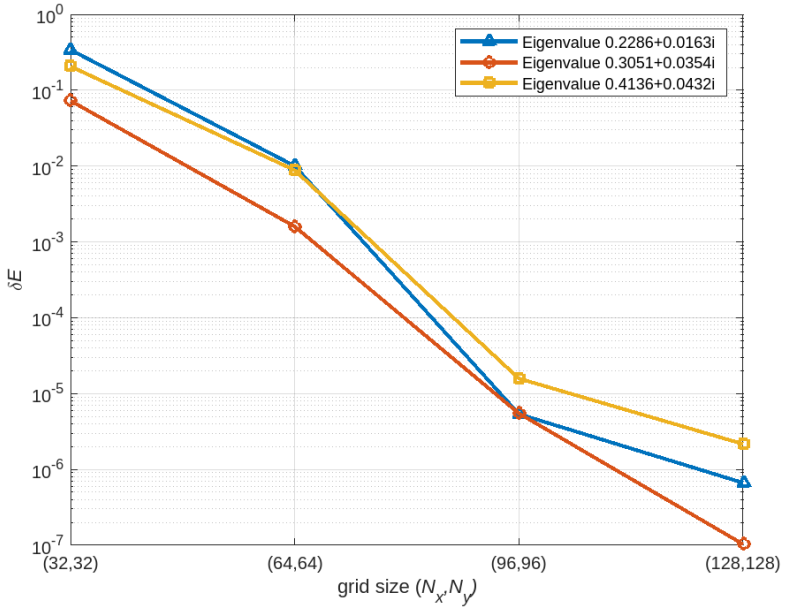


Figure 5.13: Relative spectral discretization error δE as the function of the grid size (N_x, N_y).

6

Discussion and Conclusion

As we approach the culmination of our thesis project, we embark on the final chapter, which entails a review and summary of the entire undertaking. This concluding chapter is divided into two main sections: the discussion and the conclusion. The discussion section critically examines the outcomes and shortcomings of our research, delves into the potential applications of the project, and explores what can be done in the future. And then as the conclusion of this master's thesis, we draw our final remarks. With the final conclusion, we mark the end of our journey through this thesis project while anticipating the new horizons that lie ahead.

6.1. Discussion

In this section, we engage in a comprehensive discussion that encompasses diverse facets of our research, including the conclusion of research questions, limitations, potential applications, and avenues for future exploration. This discussion section serves as a critical reflection on the outcomes of our research.

6.1.1. Answers to research questions

We have successfully addressed all five research questions through the preceding chapters on research methodology and numerical results. We now proceed to summarize the conclusions for each of the questions:

1. Discretization of three-body equations

In Chapter 2, we presented the pseudo-spectral discretization method and demonstrated its application in extending the two-body problem to the three-body problem using the Kronecker product. However, due to the inclusion of the Siegert boundary conditions, a straightforward application of the Kronecker product to the two-body discretization, as proposed in [27], is not feasible. Subsection 2.2.2 showed it necessary to adjust the discretization approach in order to maintain the tensor structure of the operators while accommodating the boundary conditions.

2. Comparison of eigensolvers

In Subsection 5.1.2, we conducted a performance comparison between two eigensolvers: the Jacobi-Davidson QZ algorithm and the quadratic Jacobi-Davidson algorithm. The JDQZ approach is utilized for solving the linearized eigenvalue problem, while the quadratic JD algorithm directly addresses the QEP. Our numerical results demonstrate that although JDQZ and quadratic JD exhibit similar convergence speeds and scalability, the computational time required by JDQZ is significantly longer. Furthermore, the linearization technique applied to the three-body problem disrupts the tensor structure of the operators, rendering the exploitation of the tensor product scheme challenging during parallelization. Consequently, the quadratic Jacobi-Davidson algorithm emerges as a more suitable method for addressing three-body problems.

3. Preventing Repetition of Solutions

In order to circumvent the repetition of convergence to the same eigenpairs within the Jacobi-Davidson iterations, we employed a strategy of incorporating all the converged eigenvectors into the search space, which we term ”locking”. Our numerical results have validated the efficacy of the ”locking” technique. Additionally, besides the ”locking” approach, we introduced a selection criterion proposed by [17] in Appendix A. This criterion is designed to selectively filter out converged eigenpairs among all potential candidates.

4. Preconditioning

In the context of addressing three-body problems, we devised a preconditioning technique for enhancing the convergence speed of the quadratic Jacobi-Davidson method. The preconditioner draws inspiration from the one employed by [27]. The preconditioner is composed of the shifted stiffness and damping matrices, both of which retain the tensor structure. This tensor structure enables us to efficiently perform preconditioning by solving the Sylvester equation. As demonstrated in Subsection 4.2.2, our preconditioning method is highly effective in significantly accelerating the convergence speed of the Jacobi-Davidson iterations, resulting in substantial time savings.

5. Parallelization

To tackle the great computational demands inherent in three-body problems, we developed a parallel Jacobi-Davidson algorithm on the supercomputer DelftBlue. Chapter 4 provides an overview of the conceptual framework behind the implementation of the parallel JD algorithm. The adoption of a tensor product scheme serves to reduce the computational complexity of operator application. Our numerical experiments were restricted to a maximum of 64 compute cores, given that the problem dimensions remained relatively modest, and thus the problem remained distant from being compute-bound. Subsection 5.2.2 delves into an analysis of the speedup achieved by the parallel program. Our findings affirm that, for smaller problem sizes, employing an excessive number of compute cores does not yield discernible benefits in terms of reducing computational time.

6.1.2. Limitations

While our findings have yielded promising results, it is crucial to acknowledge that no research endeavor is without its constraints. In the previous chapter, we presented a demonstration of the convergence behavior of the absolute/relative error of resonance poles in two/three-body problems. It is also intuitive that both the absolute error and the relative spectral error drop as the number of grid points increases. However, we have never yet addressed another critical factor that can significantly impact the accuracy of the numerical results: the cutoff.

Indeed, besides the grid points, the domain length can also influence the accuracy of numerical results. As discussed in the introductory chapter, when computing resonant states, it is crucial to impose a sufficiently large cutoff on the domain. This ensures that the potentials approach approximately zero at the boundary points. However, a longer cutoff does not necessarily guarantee higher precision. To prove the statement, we again conducted a numerical experiment on the high-barrier Pöschl-Teller two-body system. The numerical results shown in Fig. 6.1 provide straightforward evidence. We compare the numerical solutions with the analytic resonance poles $2 \pm 0.5i$ in the case of a high barrier ($\lambda = \frac{1}{2} + 2i$). It can be observed from Fig. 6.1 that the algorithm achieves the lowest error with a cutoff length of 15 when the grid is finer. Neither excessively long nor too short cutoff lengths result in better accuracy. Similar observation also applies to more general two/three-body problems, such as those involving Gaussian potentials.

Now the question remains: What is the optimal length for the cutoff? Additionally, what is the best combination of the number of grid points and cutoff length for achieving the highest accuracy? Unfortunately, our research has not encompassed this study due to the limited scope of our investigation. However, exploring the influence of cutoff lengths and grid points on the accuracy of numerical results

in the context of more general two/three-body problems with Gaussian or Lorentzian potentials could be a promising avenue for future research.

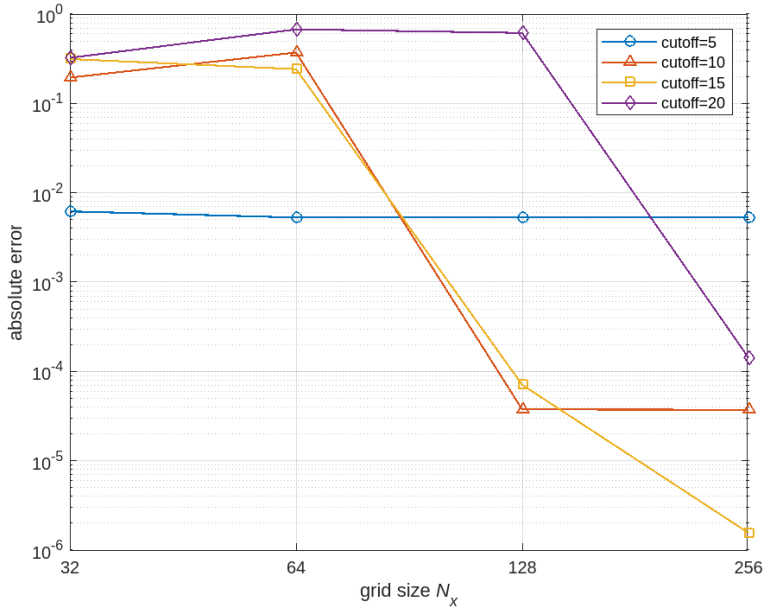


Figure 6.1: Absolute error between the numerical results and analytical solutions for the resonance poles $2 \pm 0.5i$ with varying cutoff lengths.

Another noteworthy aspect of the research relates to parallelization. The most noteworthy instance is the preconditioning step. We employ the *?trsyl* function from LAPACK for preconditioning, which inherently operates sequentially. Consequently, every preconditioner application within our parallel Jacobi-Davidson program is executed sequentially. Notably, it is indeed viable to harness shared-memory parallelism within LAPACK. However, as of the completion of this thesis, the parallel capabilities of *?trsyl* have not been exploited, which is a regrettable circumstance. We intend to rectify this limitation by updating the parallel LAPACK function after the conclusion of this article.

The final deficiency pertains to the numerical experiments of parallel JD. In the final subsection of Chapter 4, it was discussed that our parallel program supports both shared and distributed memory paradigms through the utilization of OpenMP and MPI. However, the benefits of using more than one compute node on multiple distributed memory are limited. This limitation arises from the relatively small vector size, allowing thousands of vectors to be accommodated within the memory of a single compute node in DelftBlue. As a result, the 1D three-body problem solved in this study remains significantly below the compute-bound. Indeed, in our numerical experiments, we limit the number of grid points to only 256 along both the x and y coordinates. Due to this relatively small size of the problem, the total computational effort required by the 1D three-body problems in our experiments is a memory-bound problem and does not necessitate nor yield significant benefits from utilizing supercomputing resources. When our study is extended to higher dimensions, such as 2D three-body problems, the utilization of supercomputing becomes more essential and potent.

6.1.3. Applications

The primary objective of this thesis is to develop an eigensolver capable of efficiently solving the quadratic eigenvalue equations arising from quantum two/three-body problems. Throughout the preceding chapters, we have presented comprehensive evidence supporting the convergence capabilities of our Jacobi-Davidson algorithm as the grid resolution is successively increased when applied to the QEPs derived from pseudo-spectral discretization of few-body problems. In this thesis, our focus has been on utilizing Gaussian potentials to describe the interactions between quantum particles when assessing and testing our eigensolvers. However, the numerical algorithm we have developed is of course not limited to Gaussian potentials alone. It possesses the capability to handle various other potential options, including Lorentz potentials and even more complex potential forms, such as the

Pöschl-Teller potential.

The versatility of the Jacobi-Davidson method extends beyond its application in computing resonant states of quantum few-body systems with various potential terms. It can also be employed in a wide range of other fields that involve quadratic eigenvalue problems. Quadratic eigenvalue problems arise in diverse scientific and engineering disciplines, including the dynamic analysis of structural mechanical and acoustic systems, electrical circuit simulation, fluid mechanics, and modeling microelectronic mechanical systems [28]. The ability of our eigensolver to tackle such problems opens up possibilities for its utilization in these domains. In particular, when the objective is to obtain only a limited number of specific eigenvalues, our Jacobi-Davidson algorithm has demonstrated superior efficiency compared to the conventional combination of linearization and linear eigensolvers.

Our C++ implementation of the parallel Jacobi-Davidson algorithm provides the opportunity to solve large-scale quadratic eigenvalue problems on supercomputers with multi-core CPUs. Although our program was initially designed for solving QEPs specifically arising from 1D three-body resonant problems, it can be extended beyond this domain. The tensor product scheme we introduced can be extended to tackle even higher-dimensional few-body problems, such as 2D three-body problems. By furnishing an application programming interface to input stiffness, damping, and mass operators, our implementation can be adapted to their solution. Our vision is to mold our parallel JD into a comprehensive computational toolkit, providing ease of use for researchers and practitioners in the field of quantum physics to proficiently address large-scale eigenvalue problems stemming from few-body phenomena.

6.1.4. Future work

The whole project is developed under the context of a physical problem: the computation of resonant states of quantum few-body systems. However, the emphasis lies on the development of the solver rather than delving into physics. The thesis does not entail an exploration of specific physics models. Instead, we furnish a comprehensive methodology and toolset geared toward solving challenges in few-body resonance problems. Consequently, our forthcoming endeavors can be structured around applying our algorithm to address actual few-body models. This progression could potentially lead us to uncover novel phenomena in the realm of quantum physics.

Our current work has focused on computing resonances of quantum two-body and 1D three-body systems, where the quantum particles are confined to a single line. However, there is potential for future extensions of this research to encompass two-dimensional three-body problems, where the three particles are located on the same plane. In two-dimensional three-body problems, there are a total of four coordinates (or dimensions), where $\vec{x}$ and $\vec{y}$ each have two dimensions. We can extend our problem to higher (more than 2) dimensions by utilizing Kronecker products again, as demonstrated in the work of [27]. However, it is worth noting that the discretization of three-body problems in two spatial dimensions needs to be carefully checked due to the presence of boundary conditions. Apart from the resonant states investigated in this article and the bound states studied by [27] in few-body problems of low dimension, the methodology developed can be further extended to explore additional states of three-body problems in 1D and 2D. This includes virtual states and bound states that are embedded within the continuum.

In addition to the study of quantum few-body problems, our next step involves expanding the algorithm we have developed thus far to serve a broader range of purposes. The current focus of our high-performance implementation of the QEP solver that we have developed is on computing quantum few-body problems exclusively. However, as indicated in Subsection 6.1.3, quadratic eigenvalue problems have been observed to arise in various fields. By simply modifying the input operators K, C, and M, our solver can be adapted to address QEPs derived from other problem domains. Moreover, at present, our program exclusively supports parallelism on multi-core CPU systems with shared or distributed memory architecture. There is potential for enhancing our program's functionality to encompass heterogeneous computing platforms, such as GPUs.

6.2. Conclusion

The title of this master's thesis, "Computing Resonant States of a Quantum Mechanical Three-body Problem on Supercomputers," may appear to be primarily focused on a physical topic. However, it is important to clarify that the project's main emphasis lies in the development of a robust mathematical methodology within the context of the quantum three-body problem. While the subject matter may have physical implications, the primary goal is to advance the numerical methods involved in studying resonant states in this quantum mechanical system with the help of high-performance computing.

Throughout the development of the numerical solver, which includes both discretization and the quadratic eigensolver, we have diligently addressed the research questions that were initially proposed during the literature review stage. In Chapter 2, we provide insights into formulating test problems by studying pseudo-spectral discretization. Through a thorough exploration of implementing the Kronecker product within the framework of pseudo-spectral discretization, we have successfully addressed the initial research question concerning the discretization of three-body problems. In Chapters 3 and 5, we address questions related to the eigensolvers (questions 2, 3, and 4), where we introduce the mathematical aspects of the eigensolvers and analyze their performance, respectively. Furthermore, the final research question, regarding the parallelization of the JD algorithm, is comprehensively examined in Chapter 4 and is specifically discussed in Subsection 5.2.2. In these sections, we provide an intricate account of the employed parallelization methodologies, along with presenting the numerical outcomes illustrating the performance enhancements achieved through parallel processing.

Throughout the entire project, we have meticulously developed a comprehensive algorithmic toolkit based on the Jacobi-Davidson eigensolver. This toolkit is adept at solving the quadratic eigenvalue problem arising from the two/three-body resonant systems. The numerical results presented in this study demonstrate the effectiveness of our JD algorithm in successfully converging to several resonance poles of quantum two/three-body problems. However, it is important to note that the few-body problems we have addressed thus far have not been directly applied to any real physical model. As the primary focus of this project lies in the development of the computational methodology, we have not delved extensively into the physical implications of the results.

It is promising to state that a parallel quadratic Jacobi-Davidson eigensolver has been successfully developed to solve the quantum 1D three-body problems. However, the study does not conclude here. In the future, the next step is to apply the solver to a real physical model, producing results that hold genuine physical meaning. Furthermore, we can expand the scope of research to encompass 2D three-body problems, which involve four dimensions. 2D three-body problems introduce novel challenges and raise pertinent questions including discretization, preconditioning, parallelization, and so on. Furthermore, from a software engineering perspective, we aspire to augment the hybrid nature of our parallelization approach, enabling our algorithm to efficiently leverage heterogeneous computing platforms, such as GPUs.

References

[1] Delft High Performance Computing Centre (DHPC). *DelftBlue Supercomputer (Phase 1)*. 2022. URL: https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1.

[2] P. Arbenz. *Chapter 12 The Jacobi-Davidson Method, lecture notes, Numerical Methods for Solving Large Scale Eigenvalue Problems 252-0504-00G, ETH Zürich*. 2018.

[3] R. Baltensperger and J.-P. Berrut. "The errors in calculating the pseudospectral differentiation matrices for Chebyshev-Gauss-Lobatto points". In: *Computers & Mathematics with Applications* 37.1 (1999), pp. 41–48.

[4] R. Bartels and G. Stewart. "Solutions of the matrix AX+BC = C". In: *Communications of the ACM* 15 (1972), pp. 820–826.

[5] H.A. Bethe and E.E. Salpeter. *Quantum mechanics of one-and two-electron atoms*. Springer Science & Business Media, 1957.

[6] J.G.L. Booten et al. *A preconditioned Jacobi-Davidson method for solving large generalized memory machines*. Tech. rep. 1994.

[7] L.J. Boya. "Quantum-mechanical scattering in one dimension". In: *La Rivista del Nuovo Cimento* 31.2 (2008), pp. 75–139. ISSN: 1826-9850. DOI: 10.1393/ncr/i2008-10030-4.

[8] J.P. Boyd. *Chebyshev and Fourier spectral methods*. New York: Dover, 2000.

[9] D. Cevik et al. "Resonances and antibound states of Pöschl-Teller potential: Ladder operators and SUSY partners". In: *Physics Letters A* 380.18-19 (2016), pp. 1600–1609.

[10] C. Effenberger. "Robust Successive Computation of Eigenpairs for Nonlinear Eigenvalue Problems". In: *SIAM Journal on Matrix Analysis and Applications* 34.3 (2013), pp. 1231–1256.

[11] D.R. Fokkema, G.L.G. Sleijpen, and H.A. der Vorst. "Jacobi–Davidson Style QR and QZ Algorithms for the Reduction of Matrix Pencils". In: *SIAM Journal of Scientific Computing* 20.1 (1998), pp. 94–125.

[12] M.B. van Gijzen. "The parallel computation of the smallest eigenpair of an acoustic problem with damping". In: *International Journal for Numerical Methods in Engineering* 45.6 (1999), pp. 765–777.

[13] G.H. Golub and C.F. van Loan. *Matrix Computations*. 3rd ed. Johns Hopkins University Press, 1996.

[14] L. Happ, M. Zimmermann, and M. Efremov. "Universality of excited three-body bound states in one dimension". In: *Physical Review A* 100.1 (2019).

[15] N. Hatano, T. Kawamoto, and J. Feinberg. "Probabilistic interpretation of resonant states". In: *Pramana* 73.3 (2009), pp. 553–564.

[16] N. Hatano et al. "Some properties of the resonant state in quantum mechanics and its computation". In: *Progress of Theoretical Physics* 119.2 (2007), pp. 187–222.

[17] M.E. Hochstenbach and B. Plestenjak. "Computing several eigenvalues of nonlinear eigenvalue problems by selection". In: *Calcolo* 57.2 (2020), p. 16.

[18] C.W. Hsu et al. "Bound states in the continuum". In: *Nature Reviews Materials* 1.9 (2016), p. 16048.

[19] H. Igel. "Wave propagation in three-dimensional spherical sections by the Chebyshev spectral method". In: *Geophysical Journal International* 136.3 (1999), pp. 559–566.

[20] N. Moiseyev. "Derivations of universal exact complex absorption potentials by the generalized complex coordinate method". In: *Journal of Physics B: Atomic, Molecular and Optical Physics* 31.7 (1998), pp. 1431–1441.

[21] P. Naidon and S. Endo. “Efimov physics: a review”. In: *Reports on Progress in Physics* 80.5 (2017), p. 056001.

[22] M. Razavy. “Gamow’s Theory, Complex Eigenvalues, and the Wave Function of a Decaying State”. In: *Quantum Theory of Tunneling*. Singapore: World Scientific, 2003. Chap. 5, pp. 53–72.

[23] K. Richter, G. Tanner, and D. Wintgen. “Classical mechanics of two-electron atoms”. In: *Physical Review A* 48.6 (1993), pp. 4182–4196.

[24] B. Shizgal. *Spectral methods in chemistry and physics*. Dordrecht: Springer, 2015.

[25] G.L.G. Sleijpen, M.B. van Gijzen, and H.A. van der Vorst. “Quadratic eigenproblems are no problem”. In: *SIAM News* 29.7 (1996), pp. 8–9.

[26] G.L.G. Sleijpen and H.A. der Vorst. “A Jacobi-Davidson Iteration Method for Linear Eigenvalue Problems”. In: *SIAM Review* 42.2 (2000), pp. 267–293.

[27] J. Thies et al. “Tensor product scheme for computing bound states of the quantum mechanical three-body problem”. In: *Journal of Computational Science* 64 (2022), p. 101859.

[28] F. Tisseur and K. Meerbergen. “The Quadratic Eigenvalue Problem”. In: *SIAM Rev.* 43.2 (2001), pp. 235–286.

[29] O.I. Tolstikhin, V.N. Ostrovsky, and H. Nakamura. “Siegert pseudostate formulation of scattering theory: One-channel case”. In: *Physical Review A* 58.3 (1998), pp. 2077–2096.

[30] L.N. Trefethen. *Spectral Methods in MATLAB*. Philadelphia: Society for Industrial and Applied Mathematics, 2000.

[31] H. Voss. “A Jacobi–Davidson method for nonlinear and nonsymmetric eigenproblems”. In: *Computers & Structures* 85.17 (2007), pp. 1284–1292.

[32] T. Walsh and D. Day. *Quadratic eigenvalue problems.* Tech. rep. Albuquerque, NM, and Livermore, CA (United States): Sandia National Laboratories (SNL), 2007.

A

Selection Criterion of the Jacobi-Davidson Method

M.E. Hochstenbach and B. Plestenjak introduce a special selection criterion [17] for the computation of multiple eigenvalues of nonlinear eigenvalue problems. While our research did not yield successful results in utilizing selection to prevent recurring convergence of eigenvalues, the methodology remains noteworthy. Before presenting the selection criterion, let's first define the divided difference for the polynomial $T(\lambda) = K + \lambda C + \lambda^2 M$:

Definition 9. *The divided difference for $T(\lambda)$ is defined as*

$$T[\lambda, \mu] := \begin{cases} \dfrac{T(\lambda) - T(\mu)}{\lambda - \mu}, & \lambda \neq \mu \\ T'(\lambda) \quad , & \lambda = \mu. \end{cases}$$

Apart from the divided difference, another essential ingredient for the selection process is obtaining the left eigenvectors corresponding to the converged eigenvalues. To compute the left eigenvectors, we employ an iterative algorithm designed to solve $T(\lambda)^* \mathbf{q} = 0$ for $\mathbf{q}$.

Algorithm 6 An iterative algorithm for computing a null vector of a singular matrix

Input: Singular matrix A, (random) nonzero starting vector $\mathbf{q}_0$.
Output: An approximate null vector $\mathbf{q}$ of A with $\|A\mathbf{q}\| \leq$ tol.
Compute $\mathbf{b} = A\mathbf{q}_0 / \|A\mathbf{q}_0\|$
Solve approximately $A\mathbf{x} = \mathbf{b}$ with an iterative method
Set $\mathbf{q} = (\mathbf{x} - \mathbf{q}_0)/\|\mathbf{x} - \mathbf{q}_0\|$

In the context of the Jacobi-Davidson algorithm, we assume that we have already computed d eigenvalues and corresponding right and left eigenvectors $(\lambda_1, \mathbf{p}_1, \mathbf{q}_1), \cdots, (\lambda_d, \mathbf{p}_d, \mathbf{q}_d)$. Suppose that $(\theta, \mathbf{u})$ is a candidate approximation for the next eigenpair. The selection rule is given by

$$\max_{i=1,\cdots,d} \frac{|\mathbf{q}_i^* T[\lambda_i, \theta]\mathbf{u}|}{|\mathbf{q}_i^* T'(\lambda_i)\mathbf{p}_i|} < \eta, \tag{A.1}$$

where $0 < \eta < 1$ is a fixed constant which controls the strictness of the selection. With the selection criterion (A.1), we present a modified version of the algorithm 4 denoted as algorithm 7.

Algorithm 7 does not introduce significantly greater complexity in its implementation when compared to algorithm 4. In contrast to "locking", it is not necessary to keep converged eigenvectors in the search space, so that the entire search space may be devoted to new information. During the selection process, there is a possibility that none of the Ritz vectors meet the selection criterion. In such an event, we proceed by selecting the best Ritz pair based on the target eigenvalue, disregarding the selection

criteria. Furthermore, to prevent the method from rediscovering the same eigenpair, the step of checking whether the residual norm is smaller than the tolerance is skipped. The underlying concept behind this approach lies in the expansion of the subspace, where Ritz approximations for other eigenvalues are expected to emerge during the extraction process.

Algorithm 7 The Jacobi-Davidson algorithm for computing several eigenpairs of the quadratic eigenvalue problem $\lambda^2 M\mathbf{p} + \lambda C\mathbf{p} + K\mathbf{p} = 0$ using selection

Input: initial vector t. Normalize **t** in M-norm
$V_0 = [\,]$, $m = 0$
while number of eigenvalues desired not met **do**
  $m = m + 1$, $V_m = rgs(V_{m-1}, \mathbf{t})$ ▷ orthogonalization
  Compute KV_m, CV_m, MV_m
  Compute the projected matrices $H_K = V_m^* KV_m, H_C = V_m^* CV_m$
  Compute approximation to the desired eigenvalue θ and corresponding eigenvector
  s of the projected problem $\theta^2\mathbf{s} + \theta H_C\mathbf{s} + H_K\mathbf{s} = 0, \|\mathbf{s}\| = 1$ ▷ Rayleigh-Ritz
  $\mathbf{u} = V_m\mathbf{s}$
  Select the best pair $(\theta, \mathbf{u})$ among pairs satisfying criterion (A.1) ▷ selection
  $\mathbf{w} = 2\theta M\mathbf{u} + C\mathbf{u}$, $\mathbf{r} = \theta^2 M\mathbf{u} + \theta C\mathbf{u} + K\mathbf{u}$
  if $\|\mathbf{r}\| <$tol and (A.1) satisfied **then**
    Accept the approximate eigenpair $(\theta, \mathbf{u})$
    Choose the next-best candidate pair $(\theta, \mathbf{u})$ satisfying (A.1)
    Compute the residual **r**
  end if
  if $m = m_{max}$ **then** ▷ restart
    $V_{m_{min}} = V_m S_{:,1:m_{min}}$, where $S_{:,1:m_{min}}$ is composed of m_{min} eigenvectors computed in Rayleigh-Ritz step whose corresponding eigenvalues are closest to the target τ and satisfy criterion (A.1).
  end if
  (Approximately) solve the correction equation for t (possibly with conditioner)
  $(I - (\mathbf{w}\mathbf{u}^*)/(\mathbf{u}^*\mathbf{w}))(\theta^2 M + \theta C + K)(I - (\mathbf{u}\mathbf{u}^*)/(\mathbf{u}^*\mathbf{u}))\mathbf{t} = -\mathbf{r}$, $\mathbf{t} \perp \mathbf{u}$ ▷ correction
end while