New Preconditioned Conjugate Gradient Methods for Some Structured Problems in Physics

---

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Mathematical Science

---

by
Tianqi Zhang
December 2023

Accepted by:
Dr. Fei Xue, Committee Chair
Dr. Leo G. Rebholz
Dr. Timo Heister
Dr. Yuyuan Ouyang

Abstract

This dissertation concerns the development and analysis of new preconditioned conjugate gradient (PCG) algorithms for three important classes of large-scale and complex physical problems characterized by special structures. We propose several new iterative methods for solving the eigenvalue problem or energy minimization problem, which leverage the unique structures inherent in these problems while preserving the underlying physical properties. The new algorithms enable more efficient and robust large-scale modeling and simulations in many areas, including condensed matter physics, optical properties of materials, stabilities of dynamical systems arising from control problems, and many more. Some methods are expected to be applicable to a broader range of applications. For instance, the frameworks of the PCG method presented in Chapter 3 and 4 can be extended to address various types of BEC and potential energy minimization problems. Additionally, the Chebyshev M-LOBPCG method introduced in Chapter 5 can be expanded to tackle symmetric eigenvalue problems.

In Chapter 3, we develop two numerical methods for computing the steady-state solutions of Allen-Cahn (AC) and Cahn-Hilliard (CH) equations: the PCG method and Picard iterative method with Anderson acceleration (AA). The PCG method is developed from an optimization perspective, which is equivalent to minimize the discrete energy $E(u)$. We propose a simple approach for fast evaluation of the energy functional, which enables the exact line search. We introduce two preconditioners for the PCG method, which can be applied efficiently in the Fourier pseudo-spectral discretization scheme. Moreover, we construct and analyze two Picard iterations to solve the steady-state equations of AC and CH, where the inexact AA method is applied to speed up the convergence. We compare these two methods with the second order exponential time differencing schemes (ETDRK2) [37,60]. We conclude that the PCG method outperforms the other two methods, while avoiding tuning the parameters.

In Chapter 4, we propose a new nonlinear PCG method in real arithmetic for computing the ground states of rotational Bose-Einstein condensate (BEC), modeled by the Gross–Pitaevskii equation (GPE). Our algorithm presents a few improvements of the PCG method in complex arithmetic studied by Antoine, Levitt, and Tang [J. Comput. Phys., 343 (2017), pp. 92-109]. We show that the special structure of the energy functional $E(\phi)$ and its gradient with respect to ϕ can be fully exploited in real arithmetic to evaluate them more efficiently in linear algebra operations. We propose a simple approach for fast evaluation of the energy functional at many different step sizes along the search direction at little computational cost, which enables a variety of line search methods including exact line search. Most importantly, we derive the discrete Hessian operator of the energy functional and propose a shifted Hessian preconditioner for PCG, with which the ideal preconditioned Hessian has favorable eigenvalue distributions independent of the mesh size. This suggests that PCG with our ideal Hessian preconditioner is expected to exhibit mesh size-independent asymptomatic convergence behavior. In practice, our preconditioner is constructed by incomplete Cholesky factorization of the shifted discrete Hessian operator based on high-order finite difference discretizations, updated periodically with the progress of PCG iterations. Numerical experiments in 2D and 3D domains show the efficiency of fast energy evaluation, the robustness of exact line search, and the improved convergence of PCG with our new preconditioner in iteration counts and runtime, notably for more challenging rotational BEC problems with high nonlinearity and rotational speed.

In Chapter 5, we study the iterative methods for the Bethe-Salpeter eigenvalue (BSE) problem. The discretized BSE problem arises in many body physics and quantum chemistry. Discretization leads to an algebraic eigenvalue problem involving a matrix $H \in \mathbb{C}^{2n\times 2n}$ with a Hamiltonian-like structure. After appropriate transformations, we show that the real BSE eigenproblem of form I and the complex BSE eigenproblem of form II can be transformed into real product eigenvalue problems of order n and $2n$, respectively. We propose a variant of the locally optimal block preconditioned conjugate gradient (LOBPCG) based on polynomial filters to improve the robustness and effectiveness of a few well-known existing algorithms for computing the lowest eigenvalues of the product eigenproblems. We show that our ideal locally optimal algorithm delivers Rayleigh quotient approximation to the desired lowest eigenvalue that satisfies a global quasi-optimality, which is similar to the global optimality of the preconditioned conjugate gradient method for the iterative solution of a symmetric positive definite linear system. The robustness and efficiency of the proposed method is illustrated by numerical experiments.

Dedication

To the ones who love me and the ones I love :

It is time to see you again....

Acknowledgments

I walked for a long time to get here. It is time to say thanks.

First of all, I would like to give my heartfelt thanks to my supervisor, Dr. Fei Xue, for his kindness, illuminating guidance and profound knowledge. He taught me the first course in scientific computing. From his insightful and lively lectures, I have obtained a better understanding of scientific computing and thus developed interest in numerical linear algebra. During my research period, he told me what high-quality research is and how to conduct research, which led me to the world of research. He is not only an excellent advisor, but also a nice friend. Without his keen insights and constant encouragement, this dissertation would not have been finished.

Secondly, I must express my appreciation to all professors who have taught and guided me in the School of Mathematical and Statistical Sciences at Clemson University. Without their help, I could not have continued on the path of studying mathematics and statistics. Especially, I would like to express my gratitude to my master advisor Dr. Qiong Zhang and my dissertation committee members Dr. Leo Rebholz, Dr. Timo Heister and Dr. Yuyuan Ouyang for their great support and invaluable advice. They are the best professors I've ever met.

Finally, I would like to give my heartfelt thanks to my parents and my friends, for their endless love and care for me. Whatever I need and wherever I go, they are always there supporting me without any requirement in return. I remember the days you love me, for a very long time. I keep running just to catch up with myself who was highly expected.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The conjugate gradient method is an iterative algorithm to solve systems of linear equations, particularly symmetric positive definite matrices. It was first introduced by Magnus Hestenes and Eduard Stiefel [55] in 1952, and then became one of the most popular and widely used iterative methods for solving large-scale linear systems such that

$$Ax = b, \tag{1.1}$$

where $x \in \mathbb{R}^n$ is an unknown vector, $b \in \mathbb{R}^n$ is a known vector, and $A \in \mathbb{R}^{n \times n}$ is a known, symmetric, positive definite (or positive-indefinite) matrix. The method gained popularity due to its efficiency in solving large-scale sparse linear systems, which are common in various scientific and engineering applications. Researchers have made various extensions and improvements to the conjugate gradient method in many fields. Nowadays, it still remains an important algorithm in numerical linear algebra and optimization.

The conjugate gradient (CG) methods comprise a class of unconstrained optimization algorithms, which are well suited for large-scale problems due to the simplicity of their iterations and low memory requirements. Moreover, the CG methods exhibit favorable local and global convergence properties. The nonlinear conjugate gradient method is designed to solve the following unconstrained optimization problem

$$\min f(x) : x \in \mathbb{R}^n \tag{1.2}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a continuously differentiable function, bounded from below. In practice, $f(x)$ may or may not be convex. Starting from an initial guess x_0, it generates a sequence x_k $(k = 1, 2, \cdots)$, using the recurrence

$$x_{k+1} = x_k + \alpha_k d_k, \tag{1.3}$$

where α_k is the step size obtained by a line search and d_k is the search direction such that

$$d_{k+1} = -g_{k+1} + \beta_k d_k, \text{ and } d_0 = -g_0. \tag{1.4}$$

Here, β_k is the update CG parameter and $g_k = \nabla f(x_k)$, i.e., g_k is the gradient of $f(x)$ evaluated at x_k. Different CG methods correspond to different choices for the scalar β_k. Note that the application of the conjugate gradient method may vary depending on the specific problem and its formulation.

The conjugate gradient method can also be applied to solve eigenvalue problems, specifically to find a few eigenvalues and eigenvectors of a symmetric positive definite matrix. Consider a symmetric eigenvalue problem

$$Ax = \lambda x, \tag{1.5}$$

where $A \in \mathbb{R}^{n \times n}$ is symmetric and positive definite and $(\lambda,\ x)$ is the corresponding eigenpair. In this case, finding the smallest eigenvalue and the corresponding eigenvector of A is equivalent to minimizing its Raleigh quotient $\frac{x^T Ax}{x^T x}$. A well known variation of the CG method is called the locally optimal block preconditioned conjugate gradient (LOBPCG) method, which is used to compute a few smallest eigenvalues and the corresponding eigenvectors of a symmetric generalized eigenvalue problem

$$Ax = \lambda Bx \tag{1.6}$$

for a given pair (A, B) of complex Hermitian or real symmetric matrices, where the matrix B is also assumed positive-definite. Starting from an initial approximation of the desired eigenvectors

$X^{(0)} \in \mathbb{R}^{n\times k}$, the search subspace $\mathcal{S}^{(i)}$ at the ith iteration constructed by LOBPCG is

$$\mathcal{S}^{(i)} = \text{span}\{X^{(i)}, T^{-1}R^{(i)}, P^{(i)}\}, \tag{1.7}$$

where $X^{(i)}$ is the eigenvector approximation at the ith iteration, $T^{-1}R^{(i)} \in \mathbb{R}^{n\times k}$ is the preconditioned eigenresidual vectors, and the conjugate search direction $P^{(i)} \in \mathbb{R}^{n\times k}$ is defined as

$$P^{(i)} = X^{(i)} - X^{(i-1)}C_X, \tag{1.8}$$

where $C_X \in \mathbb{R}^{k\times k}$ are determined by the *primitive* Ritz vectors from the Rayleigh-Ritz procedure of last iteration.

In physics, the CG method stands out as a powerful numerical technique employed for solving linear systems of equations, optimization problems, and eigenvalue problems. The method exhibits notable effectiveness, especially in handling large systems and sparse matrices. Its extensive applications span various domains, including quantum mechanics, classical mechanics, fluid dynamics, electromagnetic field simulations, and more [3, 11, 48, 67, 74, 97, 98, 101]. Modifications or variations of the method can be strategically implemented to address specific requirements or constraints posed by the physics problem at hand. For example, in quantum mechanics, the conjugate gradient method is employed to solve the eigenvalue problem associated with the Schrödinger equation [93], which allows physicists to determine the energy levels and corresponding wave functions of quantum systems, such as atoms, molecules, and solid-state materials.

In this dissertation, we aim to develop several new preconditioned conjugate gradient methods for three important classes of large-scale and complex physical problems with special structures:

1. Computing the steady-state solutions of Allen-Cahn (AC) and Cahn-Hilliard (CH) equations.
2. Computing ground states of rotational Bose-Einstein condensate (BEC).
3. The Bethe-Salpeter Eigenvalue problem (BSE).

More importantly, it is necessary to take advantage of the special structures to design efficient problem-dependent methods that preserve the underlying physical properties of these problems. For the steady-state solutions of the AC and CH equation, a natural way is to solve the steady-state equations with some iterative nonlinear solvers, however, it could also be addressed from the

optimization perspective, since the steady-state solutions of the AC and CH equations are the *local* minimizers of the corresponding energy functional $E(u)$. Although the wave function ϕ in the BEC problem is complex-valued, we show that the special structure of the energy functional $E(\phi)$ (the objective function of minimization by PCG) and its gradient with respect to ϕ can be fully exploited in real arithmetic to evaluate them more efficiently in linear algebra operations. Based on the special structure of the BSE problems, we transform it into real product eigenvalue problems of order n and $2n$, respectively. Our proposed PCG methods are formulated based on these observations.

1.1 The steady-state solutions of Allen-Cahn and Cahn-Hilliard equations (AC/CH).

We consider the numerical solution of the steady-state of the time-scaled Allen-Cahn (AC) equation

$$\begin{cases} u_t - \gamma \Delta u + \frac{\gamma}{\epsilon^2} f(u) = 0, & (x,t) \in \Omega \times (0,T], \\ u(\cdot,t) \text{ is } \Omega\text{-periodic}, & t \in [0,T] \\ u|_{t=0} = u_0; \end{cases} \tag{1.9}$$

and the time-scaled Cahn-Hilliard (CH) equation

$$\begin{cases} u_t - \Delta\left(-\gamma \Delta u + \frac{\gamma}{\epsilon^2} f(u)\right) = 0, & (x,t) \in \Omega \times (0,T], \\ u(\cdot,t) \text{ is } \Omega\text{-periodic}, & t \in [0,T] \\ u|_{t=0} = u_0. \end{cases} \tag{1.10}$$

Here, $\Omega \subset \mathbb{R}^d$ ($d = 2, 3$) is a bounded domain, $\gamma > 0$ is the time-scaling parameter, the parameter $\epsilon > 0$ relates to the interface width of the two phases, and $f(u) = F'(u)$ with $F(u)$ being a given energy potential. We will employ the most widely used Ginzburg-Landau double-well potential in this dissertation, i.e., $F(u) = \frac{1}{4}\left(u^2 - 1\right)^2$. The steady-state AC and CH equations are

$$-\Delta u + \frac{1}{\epsilon^2} f(u) = 0 \tag{1.11}$$

and

$$-\Delta\left(-\Delta u+\frac{1}{\epsilon^2}f\left(u\right)\right)=0, \tag{1.12}$$

respectively. There are many steady-state solutions of (1.11) and (1.12) due to the non-convexity of the potential term $f(u)$. Therefore, the final solutions computed by the numerical methods most likely depend on the initial approximations we specify. Note that, for the AC equation, it is easy to check that it has three trivial solutions, i.e., $+\mathbf{1}$, $-\mathbf{1}$, and $\mathbf{0}$, which could be attained by the numerical methods. These trivial solutions also holds for the CH equation, given that the mass (1.14) of the initial approximation is $\pm|\Omega|$ or 0.

More importantly, the AC and CH equations have the same energy functional

$$E\left(u\right):=\int_{\Omega}\left(\frac{1}{2}|\nabla u|^2+\frac{1}{\epsilon^2}F\left(u\right)\right)dx. \tag{1.13}$$

The AC (1.9) and CH (1.10) equations can be viewed as the gradient flow of the energy functional (1.13) in $L^2\left(\Omega\right)$ and $H^{-1}\left(\Omega\right)$, respectively. Moreover, it is well-known that the AC equation (1.9) satisfies maximum bound principle [37] in the sense that if the initial values are pointwisely bounded by a specific constant in absolute value, then the absolute value of the solution is bounded by the same constant everywhere for all time, which does not hold true for the CH equation (1.10). However, the CH equation (1.10) conserve the following mass exactly (i.e., $M(0)=M(t)$):

$$M(t):=\int_{\Omega}u(\mathbf{x},t)d\mathbf{x}, \tag{1.14}$$

whereas the AC equation fails to conserve.

Great efforts have been devoted to the numerical analysis of AC and CH equations [30, 37, 38, 41, 43, 44, 60, 62, 84, 105, 108]. The gradient flow methods are classical methods to compute the dynamics of the AC and CH equations. A common procedure is to discretize the AC (1.9) and CH (1.10) equations in time with an appropriate scheme that satisfies the discrete energy dissipation law (energy stable, i.e., $E(u_{k+1})\leq E(u_k)$), and then solve a large system of linear equations or nonlinear equations at each time step. For example, suppose that L_p is the discrete Laplacian matrix, a stabilized first-order semi-implicit scheme for the AC equation is proposed in [84] such

that

$$\left(\frac{1}{\delta t}+\frac{S}{\epsilon^2}\right)\left(u^{n+1}-u^n\right)-L_p u^{n+1}+\frac{1}{\epsilon^2}f\left(u^n\right)=0,$$

where u^n is the approximation to $u(x,t)$ at time step $t=t_n$.This requires to solve the following linear system for u^{n+1}

$$(I-\frac{\delta t\epsilon^2}{\epsilon^2+\delta t S}L_p)u^{n+1}=u^n-\frac{\delta t}{\epsilon^2+\delta t S}f\left(u^n\right),$$

where S is a stabilizing parameter to be specified. The gradient flow methods are necessary to compute the dynamics of the AC and CH equations, however, it becomes less attractive to compute the steady state solution due to the slow convergence of the class of steepest descent methods. In order to compute the steady-state solution, the time step size specified is expected to be as large as possible such that we can attain the steady states rapidly, which might lead to stability issues for some numerical schemes. For example, for a system of ODEs $u_t=f(t,u)$, if the Jacobian $J(t,u)=\frac{\partial f}{\partial u}$ has eigenvalues of large negative parts, explicit methods all have severe restrictions of the time step sizes. This happens typically when the mesh size h is sufficiently small, and this limitation is more severe for CH equation than for AC equation, since $J(t,u)=-L_p^2+\frac{1}{\epsilon^2}L_p\,\mathrm{diag}(3u^2-1)$, whose negative largest eigenvalues behave like $\mathcal{O}(\frac{1}{h^4})$. Here, $\mathrm{diag}(3u^2-1)$ is the diagonal matrix whose diagonal entries are the elements of $3u^2-1$, where u^2 stands for the the column vector whose entries are the square of those of u. In this sense, the exponential time differencing (ETD) method [22,31,37,57,60] is great of use, which involves exact integration of the linear part of the governing equations followed by an explicit approximation of a temporal integral involving the nonlinear terms. The ETD schemes could provide satisfactory and accuracy even though the linear terms have strong stiffness, due to the exact evaluation of the contribution of the linear part. Such an advantage may lead it to compute the steady-state solutions of AC and CH equations efficiently.

1.2 Computing ground states of Bose-Einstein condensate (BEC).

Quantum theory is one of the most important science discoveries in the last century, which asserts that all objects behave like waves in the micro length scale. However, it remains hard to observe quantum phenomena in experiments due to the extremely small wavelength until the remarkable discovery of a new state of matter, the Bose-Einstein condensate (BEC). The literature on BECs has grown rapidly over the last two decades in atomic, molecular, optics, condensed matter physics, and quantum computing; see e.g., [25, 46, 47, 50, 61] and references therein. The BEC is referred to as the fifth state of matter, which was first predicted theoretically by S.N. Bose and A. Einstein, before being realized experimentally in 1995 [7, 24, 32, 35]. It is formed by cooling a gas of bosons at extremely low density to ultra low temperatures, when microscopic quantum mechanical phenomena such as the wave function interference become macroscopically apparent. In this rapidly growing research area, numerical simulation has been playing an important role in understanding the theories and the experiments. A survey of mathematical theory and numerical methods of BEC is given in [16]. At temperatures T which are much lower than the critical temperature T_c, the macroscopic behavior of a BEC can be well described by a condensate wave function ϕ which is the solution to a Gross–Pitaevskii Equation (GPE) [11]. It is extremely useful to obtain numerical solutions of such a class of equations efficiently. Calculations of stationary states, i.e. ground/excited states, and of the real-time dynamics are the most crucial problems [8, 10, 16, 45, 59]. Numerical methods for approximating the ground states are fundamental to explore the nucleation of vortices, the properties of dipolar gases, bright beams of coherent matter waves, by studying rotational, dipolar, multi-component and spinor BECs.

We consider a BEC that can be modeled by the rotational (dimensionless) GPE. In this setting, the computation of a ground state of a d-dimensional BEC takes the form of a constrained minimization problem:

$$\text{Find} \qquad \phi \in L^2(\mathbb{R}^d) \qquad \text{s.t.} \qquad \phi \in \underset{\|\phi\|=1}{\arg\min}\, E(\phi) \tag{1.15}$$

where $\|\phi\| = (\int_{\mathbb{R}^d} |\phi|^2)^{\frac{1}{2}}$ is the standard L^2-norm and $E(\phi)$ is the associated energy functional defined

as

$$E(\phi) = \int_{\mathbb{R}^d} \left[\frac{1}{2}|\nabla\phi|^2 + V(x)|\phi|^2 + \frac{\eta}{2}|\phi|^4 - \Omega\phi^* L_z \phi \right]. \tag{1.16}$$

Here, V is an external potential, η is the nonlinearity strength, Ω is the rotational speed, and $L_z = i(y\partial_x - x\partial_y)$ is the angular momentum operator.

The constrained minimization problem (1.15) can be written in the discrete form. Generation of an appropriate mesh on a finite domain $U \subseteq \mathbb{R}^d$ and application of a corresponding discretization to the continuous GPE, the ground state of BEC in discrete form is the global minimizer of the energy functional

$$E_{\eta,\Omega} = \left[-\frac{1}{2}\phi^* L_p \phi + \phi^* \mathrm{diag}(V)\phi + \frac{\eta}{2}\phi^* \mathrm{diag}(|\phi|^2)\phi - i\Omega\phi^* L_\omega \phi \right] h^d, \tag{1.17}$$

with $\|\phi\|_{\ell^2}^2 = h^d \phi^* \phi = 1$, which is a discretized version of (1.16). Here, $-L_p$ (symmetric semi-positive definite) is the negative discrete Laplacian operator, $\mathrm{diag}(V)$ and $\mathrm{diag}(|\phi|^2)$ are diagonal matrices with the values of the external trapping potential $V(\mathbf{x})$ and $|\phi(\mathbf{x})|^2$ at the mesh nodes on the diagonal, L_ω (skew symmetric) is the discrete version of the operator $y\partial_x - x\partial_y$, $\eta > 0$ denotes the repulsive particle interaction, and Ω is the angular momentum rotating speed. A direct evaluation of the gradient of the energy being zero leads to the algebraic nonlinear eigenvalue problem

$$-\frac{1}{2}L_p\phi + \mathrm{diag}(V)\phi + \eta\mathrm{diag}(|\phi|^2)\phi - i\Omega L_\omega \phi = \lambda_{\eta,\Omega}\phi, \text{with } h^d\phi^*\phi = 1, \tag{1.18}$$

where the eigenvalue $\lambda_{\eta,\Omega}$ is defined as

$$\lambda_{\eta,\Omega} = \left[-\frac{1}{2}\phi^* L_p \phi + \phi^* \mathrm{diag}(V)\phi + \eta\phi^* \mathrm{diag}(|\phi|^2)\phi - i\Omega\phi^* L_\omega \phi \right] h^d. \tag{1.19}$$

Our aim is to find the the global minimizer of (1.17) numerically. Note that the minimizer of $E_{\eta,\Omega}$ is *not* necessarily the eigenvector associated with the lowest eigenvalue of (1.18) [16].

In the literature of numerical solutions to partial differential equations (PDEs), a family of classical methods for computing the ground states of rotational BEC (1.15) and other types of BECs, or similarly the steady-state solutions to Allen-Cahn and Cahn-Hilliard equations, are based on gradient flows $u_t = -\nabla_u(E(u))$ that define a steepest descent curve $u(t)$ of the energy

$E(u)$.These algorithms have been extensively studied, most well-known and widely used for years across disciplines, with mature theoretical support. However, they require solution to a large system of linear equations at each time step, which is usually time-consuming, particularly in 3D domains with a small mesh size. In addition, these gradient flow-based methods tend to converge slowly with the progress of time steps, since they belong to the class of steepest descent methods that are notably not efficient for numerical optimization (of the energy E(u)). For example, (1.15) can be solved by the gradient flow with discrete normalization method (GFDN, also called 'imaginary time' methods in physics) [2, 9, 10, 16, 17, 20, 27, 29, 107]. More precisely, the gradient flow associated with the energy $E(\phi)$ (1.16) is

$$\partial_t \phi = -(-\frac{1}{2} L_p \phi + \mathrm{diag}(V)\phi + \eta \mathrm{diag}(|\phi|^2)\phi - i\Omega L_\omega \phi - \lambda_{\eta,\Omega}\phi), \tag{1.20}$$

which can be discretized in time with the backward-Euler discretization scheme and solved. With discrete from in time, we have

$$\frac{\widetilde{\phi}_{n+1} - \phi_n}{\Delta t} = -(-\frac{1}{2} L_p \widetilde{\phi}_{n+1} + \mathrm{diag}(V)\widetilde{\phi}_{n+1} + \eta \mathrm{diag}(|\phi_n|^2)\widetilde{\phi}_{n+1} - i\Omega L_\omega \widetilde{\phi}_{n+1} - \lambda_{\eta,\Omega}\phi_n) \tag{1.21}$$

After obtaining the solution $\widetilde{\phi}_{n+1}$ of the above equation, a projection step $\phi_{n+1} = \widetilde{\phi}_{n+1}/\|\widetilde{\phi}_{n+1}\|$ is followed to ensure the normalization constraint.

Other methods have been developed, based on numerical solution of the nonlinear eigenvalue problem [36, 100] or on optimization techniques under constraints [19, 26, 33, 34]. In the past few years, new methods have emerged, such as preconditioned conjugate gradient methods [11, 12, 94], the regularized Newton-type method [103] and the rDF-APG method [18], which seem successful but fall short of real arithmetic computation, problem-dependent preconditioning. In [11], the state-of-the-art variant of preconditioned conjugate gradient(PCG) method was proposed to solve the constrained minimization problem (1.15), which outperforms all the previous methods.

However, several issues remain in [11] to address. For example, the complex arithmetic naturally used with Fourier pseudo spectral methods does not fully exploit the special structure of rotational BEC to speed up certain basic linear algebra computations and to guarantee that the computed energy $E(\phi)$ is real. More importantly, as a most significant component of preconditioned conjugate gradient methods, the preconditioners proposed therein did not take the rotational speed

Ω into account, and the convergence rate seems to deteriorate considerably for high-speed rotational problems. In fact, the condition number of the preconditioned Hessian is shown to increase with the domain size L and h^{-2} where h is the mesh size. In addition, direct energy evaluation of $E(\phi)$ defined in (1.16) seems costly for each step size along the search direction, which entails the use of approximate line search based on quadratic approximations of $E(\phi)$ or backtracking algorithms.

1.3 The Bethe-Salpeter eigenvalue (BSE) problem.

We consider the following product eigenvalue problem

$$KMv = \lambda v \tag{1.22}$$

where K and M are real symmetric positive definite matrices. This product eigenvalue problem has several important applications. In the quantum physics, chemistry and material science communities, there is a growing demand for *ab initio* computation of the absorption spectra for molecules or surfaces of solids [66, 73, 78, 81]. This computational problem can be solved by the Bethe-Salpeter equation (BSE). The BSE approach leads to the challenging computational task of the solution of a large eigenvalue problem involving a non-Hermitian dense Hamiltonian matrix H of order $\mathcal{O}(N_b^2)$, where N_b is the the number of atomic orbitals basis set. In the past decades, many progresses have been made in structure-preserving *full diagonalization* methods [21, 51, 75, 83] to handle problems of small to medium size (e.g., up to ten thousand), limited by their arithmetic complexity of $\mathcal{O}(N_b^6)$. When a modest number of the lowest excited states of excitation energies and oscillator strengths are of interest, or when it is not clear how many excited states are needed, iterative methods are expected to have a favorable performance. Many iterative methods for solving the standard symmetric eigenvalue problem can be modified slightly and applied to compute the lowest partial spectrum of (1.22), based on the observation that KM is self-adjoint with respect to the M-inner product [98].

Depending on the exact circumstances and discretization schemes, the BSE eigenvalue prob-

lem has two forms [21]. The BSE eigenproblem of form I is defined as

$$H_1 v \equiv \begin{pmatrix} A & B \\ -B & -A \end{pmatrix} v = \lambda v, \tag{1.23}$$

where $A = A^*$, $B = B^* \in \mathbb{C}^{n\times n}$. If A and B are real symmetric matrices and $\begin{pmatrix} A & B \\ B & A \end{pmatrix} \succ 0$ (symmetric positive define), the real BSE eigenproblem of form I turns out to be the linear response eigenvalue problem (LREP). Through an orthogonal similarity transformation [14], the eigenvalue problem (1.23) can be reformulated as

$$H_1 z \equiv \begin{pmatrix} 0 & K \\ M & 0 \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = \lambda \begin{pmatrix} y \\ x \end{pmatrix}, \tag{1.24}$$

where $K = A - B$ and $M = A + B$ are $n \times n$ real symmetric positive semidefinite matrices and at least one of them is definite. As shown in [15], the LREP is equivalent to the following product eigenvalue problems:

$$KMy = \lambda^2 y \quad \text{or} \quad MKx = \lambda^2 x, \tag{1.25}$$

which is a real product eigenvalue problem of order n. This also implies that the eigenvalues of H_1 come in pairs $\{\lambda, -\lambda\}$. If the time-inversion symmetry in the basis functions is not exploited or not available, the BSE eigenproblem of form II is defined as [21]

$$H_2 v \equiv \begin{pmatrix} A & B \\ -\overline{B} & -\overline{A} \end{pmatrix} v = \lambda v, \tag{1.26}$$

where $A = A^*$, $B = B^T \in \mathbb{C}^{n\times n}$ or $\mathbb{R}^{n\times n}$. We derive a new formulation of (1.26), which is equivalent to a real product eigenvalue problem of order $2n$ and can be solved in real arithmetic. Let us define

$$J = \begin{pmatrix} 0 & I_n \\ -I_n & 0 \end{pmatrix}, \quad Q = \frac{1}{\sqrt{2}} \begin{pmatrix} I_n & -iI_n \\ I_n & iI_n \end{pmatrix}, \tag{1.27}$$

and

$$A_R = \begin{pmatrix} \mathrm{Re}(A) & \mathrm{Im}(A) \\ -\mathrm{Im}(A) & \mathrm{Re}(A) \end{pmatrix},\ B_R = \begin{pmatrix} \mathrm{Re}(B) & -\mathrm{Im}(B) \\ -\mathrm{Im}(B) & -\mathrm{Re}(B) \end{pmatrix}, \tag{1.28}$$

where Q is unitary, and A_R, B_R are real symmetric. It can be shown that H_2 is unitarily similar to a purely imaginary matrix,

$$Q^* H_2 Q = -iJ(A_R + B_R) = G. \tag{1.29}$$

For practical physical systems, we have $A_R + B_R \succ 0$ [83]. It follows that H_2 is similar to the following matrices:

$$(A_R + B_R)^{\frac{1}{2}} G (A_R + B_R)^{-\frac{1}{2}} = -i(A_R + B_R)^{\frac{1}{2}} J (A_R + B_R)^{\frac{1}{2}}, \tag{1.30}$$

which are purely imaginary Hermitian with only real eigenvalues. Moreover, since $(JH_2)^T = JH_2$, we have $(Jv)^T H_2 = v^T J^T H_2 = -v^T J H_2 = -v^T H_2^T J^T = -\lambda (Jv)^T$, which implies that Jv is a left eigenvector corresponding to the eigenvalue $-\lambda$ of H_2. Thus, the eigenvalues of H_2 also comes in pairs $\{\lambda, -\lambda\}$.

Unlike the real BSE eigenproblem of form I (LREP), which can be reformulated as (1.25), the complex BSE eigenproblem of form II (1.26) seems unlikely to be compressed to a product eigenproblem of order n. In fact, since H_2 is unitarily similar to the purely imaginary matrix $G = -iJ(A_R + B_R)$, we can solve $G^2 u = \lambda^2 u$, that is

$$G^2 u = J^T M_R J M_R u = \lambda^2 u, \tag{1.31}$$

where $M_R = A_R + B_R \in \mathbb{R}^{2n \times 2n}$ and $M_R \succ 0$. Let $K_R = J^T M_R J \succ 0$, we obtain

$$K_R M_R u = \lambda^2 u, \tag{1.32}$$

which is a real product eigenproblem of order $2n$.

Therefore, the real BSE eigenproblem of form I (LREP) and complex BSE eigenproblem of form II (CBSEP) can be transformed into a real product eigenvalue problem (1.22) of order n and

$2n$, respectively. To tackle the BSE eigenproblems, it would be sufficient to focus on the product eigenvalue problem (1.25) or (1.32). For LREP, the solution of one product eigenvalue problem yields n elements of the eigenvector associated with the original eigenvalue problem. The other n elements of the eigenvector can be easily recovered in a post-processing procedure. For CBSEP, once we obtain the lowest $2k$ eigenpairs (λ_j^2, u_j) of G^2 $(1 \le j \le 2k)$, with $\lambda_1^2 = \lambda_2^2 \le \cdots \le \lambda_{2k-1}^2 = \lambda_{2k}^2$, we can perform Raleigh-Ritz projections to construct and solve the 2×2 eigenproblems $(U_\ell^T G U_\ell, U_\ell^T U_\ell)$, where $U_\ell = [u_{2\ell-1}, u_{2\ell}]$ for $1 \le \ell \le k$ to retrieve the k lowest positive eigenvalues and eigenvectors of G, and pre-multiply those eigenvectors by Q in (1.27) to get the desired eigenvectors of H_2.

In the past decades, many iterative methods have been developed for computing the smallest eigenvalues of LREP [15, 23, 91, 92, 95, 96, 98]. Several Lanczos-type methods have been developed to solve LREP. In [95, 96], Tsiper proposed a Lanczos-type process to reduce both K and M to tridiagonal. The first Lanczos process for LREP in [90] was proposed to reduce K to tridiagonal and M to diagonal. Theoretically, Tsiper's Lanczos process converges at only half the speed of the first Lanczos process. A weighted block Golub-Kahan-Lanczos for LREP was proposed in [109], which is slightly cheaper in computational cost than the first Lanczos process. Also, we notice that a Lanczos procedure in the M-inner product for the real full BSE has been proposed in [23]. These methods have been used to build Krylov subspaces, which can be considered optimal matrix-polynomial-based methods because they obtain their solutions over the entire set of matrix polynomials of degree up to the number of iterations. For difficult problems, restarting technique may be applied to alleviate the demand for memory and computational cost, but they may cause deterioration of convergence. On the other hand, several non-Krylov subspace methods have been developed to solve LREP. A variant of LOBPCG called LOBP4DCG was proposed in [15]. An indefinite variant of LOBPCG is also proposed in [68] as a generalization of LOBP4DCG. Notice that all the variants of Lanczos and LOBPCG mentioned so far work directly on (1.24). For example, the state-of-the-art approach LOBP4DCG based on (1.24) initializes with two approximation block $X^{(0)} \in \mathbb{R}^{n\times k}$, $Y^{(0)} \in \mathbb{R}^{n\times k}$, and the two projected search subspace $\mathcal{S}_K^{(i)}$, $\mathcal{S}_M^{(i)}$ at the ith iteration constructed by LOBP4DCG are

$$\mathcal{S}_K^{(i)} = \text{span}\{X^{(i)}, T_K^{-1}R_K^{(i)}, X^{(i-1)}\}, \text{ and } \mathcal{S}_M^{(i)} = \text{span}\{Y^{(i)}, T_M^{-1}R_M^{(i)}, Y^{(i-1)}\}, \tag{1.33}$$

where $X^{(i)}$, $Y^{(i)}$ are the parts of the eigenvector approximation at the ith iteration, $T_K^{-1}R_K^{(i)}$, $T_M^{-1}R_M^{(i)} \in \mathbb{R}^{n\times k}$ are components of the projected gradient of (1.24), and T_K^{-1}, T_M^{-1} are the cho-

sen preconditioners.

In [98], the product eigenvalue problem (1.25) is proposed to be solved by LOBPCG-like methods that makes use of the M-inner product or the K-inner product. Several widely used algorithms for solving the standard symmetric eigenvalue problems can be used by simply replacing the regular Euclidean inner product with the M-inner product [98]. The M-Davidson and M-LOBPCG methods are provided therein. These PCG-type methods are usually used with preconditioning techniques to achieve fast convergence and their performance is sensitive to the quality of the preconditioners. For some eigenproblems, it might be very challenging to obtain a preconditioner of high quality. Also, if the number of desired eigenvalues is relatively large, the preconditioner becomes less effective, then it is necessary to update the preconditioner during the iterations, which involves more computational cost. To compute many smallest eigenvalues more efficiently, a Chebyshev-Davidson method was proposed in [92] based on the polynomial filter technique. Numerical results there showed that the approach could work quite well. For difficult problems, a very high degree of polynomial filter needs to be applied to make it efficient, which may lead to instability of convergence [71]. Moreover, the Davidson-type method is not storage efficient and repeated restarts could slow the convergence. Note that, the polynomial filter technique could be used in the Lanczos algorithm [42, 70] for solving the standard Hermitian eigenvalue problem, which could also work here with the M inner product. However, in order to keep the structure of the Lanczos procedure, the polynomial filter must remain fixed during the entire algorithm. Such a fixed polynomial filter requires in-depth knowledge of the spectrum to be effective, but such knowledge may not be readily available in practice. This makes these methods less robust than the Chebyshev-Davidson method that does not build Krylov subspaces and allows for the use of adaptive polynomial filters.

For large CBSEP, iterative solvers are still necessary to find a modest number of smallest positive eigenvalues, though, several direct methods leading to *full diagonalization* have been developed. Unlike the LREP, the study of structure preserving iterative methods for CBSEP still seems limited. In Chapter 5, we review two classes of non-Krylov subspace iterative methods for computing the lowest partial spectrum of the product eigenproblem (1.22). Our new method is based on a proper combination of the strengths of both algorithms.

Chapter 2

Notation and Preliminaries

In this chapter, we introduce the discretization scheme for the BEC and AC/CH problems in Chapter 3 and 4. Also, we present some preliminary results, which are useful to prove the quasi-optimality of the Chebyshev-LOBPCG method in Chapter 5.

2.1 Discretization scheme for the BEC and AC/CH problems

In this section, we introduce the discretization scheme for the BEC and AC/CH problems we employ in this dissertation. Since we consider these problems on periodic domains, the Fourier pseudo-spectral method is very accurate and widely used. Here, we introduce this discretization schemes based on the BEC problem, while it is straightforward to use it for the AC/CH problem.

The function $\phi \in L^2(\mathbb{R}^d)$ must be discretized in order to find a numerical solution of the minimization problem (1.15). Also, the discretization must be accurate enough to resolve fine details of vortexes in the solution. Several discretization schemes have been used to compute the solution to the GPE, including high-order finite difference schemes, finite element schemes with adaptive meshing strategies [33, 34], the standard pseudo-spectral schemes based on Fast Fourier Transforms (FFTs) [9–11, 17].

In the literature on numerical methods for BEC computations, the Fourier pseudo-spectral method [16] is the most widely adopted discretization. In this dissertation, we adopt the Fourier pseudo-spectral discretization scheme, which is described in 2D and its extension to other dimensions is straightforward. The wave function ϕ is truncated to a rectangular domain $[-L_x, L_x] \times$

$[-L_y, L_y]$ with periodic boundary conditions, and discretized with even number of grid points N_x, N_y in the x- and y- directions, respectively. A uniformly sampled grid is introduced: $\mathcal{D}_{N_x,N_y} := \{\mathbf{x}_{k_1,k_2} = (x_{k_1}, y_{k_2})\}_{k_1,k_2 \in \mathcal{I}_{N_x,N_y}}$, with $\mathcal{I}_{N_x,N_y} := \{0, \cdots, N_x - 1\} \times \{0, \cdots, N_y - 1\}$, $x_{k_1+1} - x_{k_1} = y_{k_2+1} - y_{k_2} = h$, and with mesh size $h = 2L_x/N_x = 2L_y/N_y$. Define the discrete Fourier frequencies (ξ_p, μ_q), with $\xi_p = p\pi/L_x$, $-N_x/2 \le p \le N_x/2 - 1$ and $\mu_q = q\pi/L_y$, $-N_y/2 \le q \le N_y/2 - 1$. The pseudo-spectral approximation $\widetilde{\phi}$ of the function ϕ in the $x-$ and $y-$directions are such that

$$\widetilde{\phi}(x,y) = \frac{1}{N_x} \sum_{p=-N_x/2}^{N_x/2-1} \widetilde{\phi_p}^*(y) e^{i\xi_p(x+L_x)},\ \widetilde{\phi}(x,y) = \frac{1}{N_y} \sum_{q=-N_y/2}^{N_y/2-1} \widetilde{\phi_q}^*(x) e^{i\mu_q(y+L_y)},$$

where $\widetilde{\phi_p}^*(y)$ and $\widetilde{\phi_q}^*(x)$ are the Fourier coefficients in the x- and y-directions, respectively; that is,

$$\widetilde{\phi_p}^*(y) = \sum_{k_1=0}^{N_x-1} \widetilde{\phi}(x_{k_1}, y) e^{-i\xi_p(x_{k_1}+L_x)},\ \widetilde{\phi_q}^*(x) = \sum_{k_2=0}^{N_y-1} \widetilde{\phi}(x, y_{k_2}) e^{-i\mu_q(y_{k_2}+L_y)}.$$

In order to evaluate the action of the discrete Laplacian and the angular rotation operators on vectors in (1.17), we also need to apply the following operators to the approximation $\widetilde{\phi}$ of ϕ, for $(k_1, k_2) \in \mathcal{I}_{N_x,N_y}$,

$$\begin{aligned}
\partial_x^2 \phi(\mathbf{x}_{k_1,k_2}) &\approx \partial_x^2 \widetilde{\phi}(x_{k_1}, y_{k_2}) = -\frac{1}{N_x} \sum_{p=-N_x/2}^{N_x/2-1} \xi_p^2 \widetilde{\phi_p}^*(y_{k_2}) e^{i\xi_p(x_{k_1}+L_x)}, \\
\partial_y^2 \phi(\mathbf{x}_{k_1,k_2}) &\approx \partial_y^2 \widetilde{\phi}(x_{k_1}, y_{k_2}) = -\frac{1}{N_y} \sum_{q=-N_y/2}^{N_y/2-1} \mu_q^2 \widetilde{\phi_q}^*(x_{k_1}) e^{i\mu_q(y_{k_2}+L_y)}, \\
x\partial_y \phi(\mathbf{x}_{k_1,k_2}) &\approx x\partial_y \widetilde{\phi}(x_{k_1}, y_{k_2}) = -\frac{1}{N_y} \sum_{q=-N_y/2}^{N_y/2-1} i x_{k_1} \mu_q \widetilde{\phi_q}^*(x_{k_1}) e^{i\mu_q(y_{k_2}+L_y)}, \\
y\partial_x \phi(\mathbf{x}_{k_1,k_2}) &\approx y\partial_x \widetilde{\phi}(x_{k_1}, y_{k_2}) = -\frac{1}{N_x} \sum_{p=-N_x/2}^{N_x/2-1} i y_{k_2} \xi_p \widetilde{\phi_p}^*(y_{k_2}) e^{i\xi_p(x_{k_1}+L_x)}.
\end{aligned}$$

Meanwhile, we also introduce the finite difference discretization scheme [58, 88], which is useful to construct the Hessian preconditioner we propose in Section 4.2. With the same uniform

mesh grids in Fourier pseudo-spectral discretization scheme, the matrices for the operators are

$$L_p = D_{2,x} \otimes I + I \otimes D_{2,y},$$
$$L_\omega = \mathrm{diag}(y_0, \cdots, y_{N_y-1}) \otimes D_x - D_y \otimes \mathrm{diag}(x_0, \cdots, x_{N_x-1}),$$

where D_x, D_y and $D_{2,x}$, $D_{2,y}$ are sparse matrices containing the coefficients of the central finite difference approximations of the first partial derivative and the second partial derivative with respect to x and y, respectively [27, 58, 88]. Also, the order of the finite difference approximation should be even numbers from 2 to 8. Note that, regardless of the discretization scheme, the discrete negative Laplacian operator $-L_p$ and the discrete angular rotation operator L_ω are real symmetric positive definite and real skew-symmetric, respectively.

2.2 Preliminaries for BSE problem

Consider the matrix pencil (MKM, M), where $K, M \in \mathbb{R}^{n\times n}$ are symmetric and positive definite. Let $\lambda_1 < \lambda_2 \leq \cdots \leq \lambda_n$ and $\{v_i\}$ be the eigenvalues and eigenvectors of the matrix pencil, such that $MKMv_i = \lambda_i M v_i$, $(v_i, v_j)_M = v_i^T M v_j = \delta_{ij}$, $\|v_i\|_M = 1$, $1 \leq i, j \leq n$.

Let x be an approximation to v_1, the eigenvector associated with the lowest eigenvalue λ_1, with the decomposition

$$x = v_1 \cos\theta + f \sin\theta, \text{where } \theta \neq 0,\ f \perp Mv_1, \text{and } \|f\|_M = 1, \tag{2.1}$$

which implies that $\|x\|_M = (x^T Mx)^{\frac{1}{2}} = (\|v_1\|_M^2 \cos^2\theta + \|f\|_M^2 \sin^2\theta)^{\frac{1}{2}} = 1$. Since $f \perp Mv_1$, it has the form $f = \sum_{j=2}^n s_j v_j$, where the scalars $\{s_j\}_{j=2}^n$ satisfy $\sum_{j=2}^n s_j^2 = \sum_{j=2}^n s_j^2 \|v_j\|_M^2 = \|f\|_M^2 = 1$. Hence, the Raleigh quotient of x is

$$\begin{aligned}\rho(x) &= \frac{x^T MKMx}{x^T Mx} = x^T MKMx \\ &= v_1^T MKMv_1 \cos^2\theta + f^T MKMf \sin^2\theta = \lambda_1 \cos^2\theta + \rho(f)\sin^2\theta,\end{aligned} \tag{2.2}$$

where $\rho(f) = \frac{f^T MKMf}{f^T Mf} = f^T MKMf = \sum_{j=2}^n s_j^2 \lambda_j \in [\lambda_2, \lambda_n]$.

Next, we'll present two important propositions of $\rho(x)$.

Proposition 2.2.1. *Let x be a vector with $\|x\|_M = 1$. The gradient and the Hessian of $\frac{1}{2}\rho(x)$ with respect to x, respectively, are*

$$\begin{aligned}
\nabla\frac{1}{2}\rho(x) &= \frac{1}{x^TMx}(MKM - \rho(x)M)x = MKMx - \rho(x)Mx, \quad \textit{and} \\
\nabla^2\frac{1}{2}\rho(x) &= \frac{1}{x^TMx}\left\{MKM - \rho(x)M - \frac{2}{x^TMx}(MKMx - \rho(x)Mx)(Mx)^T\right. \\
&\quad \left. -\frac{2}{x^TMx}Mx(MKMx - \rho(x)Mx)^T\right\} \\
&= MKM - \rho(x)M - 2(MKMx - \rho(x)Mx)(Mx)^T - 2Mx(MKMx - \rho(x)Mx)^T
\end{aligned}$$

Proof. This is done by letting $T(\rho) = \rho M - MKM$ in [89, Proposition 3.1]. □

Proposition 2.2.2. *Let x be an approximation to v_1 with the decomposition $x = v_1\cos\theta + f\sin\theta$, where $f \perp Mv_1$ and $\|f\|_M = 1$, then we have $\rho(x) - \lambda_1 = \mathcal{O}(\sin^2\theta)$, and $\|MKMx - \rho(x)Mx\| = \mathcal{O}(\sin\theta)$.*

Proof. It is easy to see that, for small θ, $\rho(x)$ is a second order approximation to λ_1, since

$$\rho(x) - \lambda_1 = \lambda_1(\cos^2\theta - 1) + \rho(f)\sin^2\theta = \sin^2\theta(\rho(f) - \lambda_1). \tag{2.3}$$

Thus, $\rho(x) - \lambda_1 = \mathcal{O}(\sin^2\theta)$. Also, the eigenresidual associated with x, is

$$\begin{aligned}
MKMx - \rho(x)Mx &= (MKM - \rho(x)M)(v_1\cos\theta + f\sin\theta) \\
&= (\lambda_1 - \rho(x))\cos\theta Mv_1 + \sin\theta(MKM - \rho(x)M)f \\
&= \sin\theta\left\{\sum_{j=2}^{n} s_j(\lambda_j - \rho(x))Mv_j - \sin\theta\cos\theta(\rho(f) - \lambda_1)Mv_1\right\}.
\end{aligned}$$

Since $\lambda_1 < \lambda_2$ and $\sum_{j=2}^{n} s_j^2 = 1$, $MKMf - \rho(x)Mf = \sum_{j=2}^{n} s_j(\lambda_j - \rho(x))Mv_j$ will not vanish as $\theta \to 0$ and $\rho(x) \to \lambda_1$. Then, for sufficiently small θ and some $\delta > 0$ independent of θ, we have

$$\begin{aligned}
(1-\delta)\sin\theta\|MKMf - \rho(x)Mf\| &\le \|MKMx - \rho(x)Mx\| \\
&\le (1+\delta)\sin\theta\|MKMf - \rho(x)Mf\|.
\end{aligned}$$

Then,

$$\|MKMx - \rho(x)Mx\| = \mathcal{O}(\sin\theta). \tag{2.4}$$

□

Chapter 3

Steady-state solutions of Allen-Cahn and Cahn-Hilliard equations

In this chapter, we study three class of numerical methods for computing the steady-state solutions of Allen-Cahn (AC) and Cahn-Hilliard (CH) equations with a Fourier pseudo-Spectral discretization method (FPSM). Firstly, we consider the Picard type nonlinear iterative solvers based on the steady-state AC (1.11) and CH (1.12) equations. We construct two contractive mappings and formulate the solvers as fixed point problems. Moreover, we apply the inexact Anderson acceleration (AA) to the Picard iterations to further accelerate the convergence. Secondly, we introduce the first and second order exponential time differencing schemes, which seems promising for solving the steady-state solution of AC/CH. We prove that, the first order scheme for CH equation satisfies the energy stability and 2-norm stability under some assumptions. Finally, we develop an efficient preconditioned conjugate gradient (PCG) method from the optimization perspective. For the PCG method, we develop a simple approach to achieve fast energy evaluations for many different step sizes along the search direction at little additional cost, which enables exact line search efficiently. We propose two efficient preconditioners, which exploits the special structure of the problem and realizes fast numerical linear algebra computations in FPSM. We compare the performances among these three methods. We find that for computing the steady-state solutions of the AC and CH

equations, the PCG method is much more efficient than the other two methods.

The remainder of this chapter is organized as follows. In Section 3.1, we develop the Picard iterative method with the Anderson acceleration. We analyze the exponential time differencing in Section 3.2. We provide a detailed description of the PCG method in Section 3.3. Section 3.4 provides numerical results in 2D and 3D domains to validate and compare the performances among these methods.

3.1 The Picard iterative method with Anderson acceleration

In order to compute the steady-state solutions of the AC and CH equations, a natural way is to solve the steady-state equations (1.11) and (1.12) with some iterative methods. In this section, we develop the Picard iterative method with inexact Anderson acceleration for computing the steady-state solutions of the AC and CH equations based on the steady-state equations (1.11) and (1.12).

We introduce the Picard iterations for the AC and CH equations as

$$u_{k+1} = \left(-\gamma L_p + \frac{\gamma}{\epsilon^2}\operatorname{diag}\left(u_k^2\right) + sI\right)^{-1}\left(s + \frac{\gamma}{\epsilon^2}\right)u_k = g^{AC}\left(u_k\right), \tag{3.1}$$

and

$$u_{k+1} = \left(\gamma L_p^2 - \frac{\gamma}{\epsilon^2}L_p\operatorname{diag}\left(u_k^2\right) + sI\right)^{-1}\left(sI - \frac{\gamma}{\epsilon^2}L_p\right)u_k = g^{CH}\left(u_k\right), \tag{3.2}$$

respectively. Here, $s > 0$ are tuning parameters to speed up the convergence. The Picard iterations (3.1) and (3.2) come from the steady-state equations (1.11) and (1.12), by replacing u with u_k or u_{k+1}. It is straightforward to check that the fixed points of $g^{AC}(u)$ and $g^{CH}(u)$ are solutions to the steady-state AC (1.11) and CH (1.12) equations, respectively. A main concern associated with the fixed point iteration is that the iterates may not converge. The contractivity of the Picard iterations is essential to ensure the convergence. To show $g^{AC}\left(u\right)$ and $g^{CH}\left(u\right)$ are contractive, it is equivalent to show that the spectral radiuses of the Jacobin matrix $J^{AC}(u)$ and $J^{CH}(u)$ are not greater than 1. The following Lemma (3.1.1) is provided in order to find the Jacobian matrices of $g^{AC}(u)$ and $g^{CH}(u)$.

Lemma 3.1.1. *Consider A, $B \in \mathbb{R}^{n\times n}$, where A is nonsingular, and $\rho\left(A^{-1}B\right) < 1$. Then, we have*

$$(A+B)^{-1} = A^{-1} - A^{-1}BA^{-1} + A^{-1}BA^{-1}BA^{-1} - \cdots . \tag{3.3}$$

Proof. The proof is straightforward. We have

$$\begin{aligned}
(A+B)^{-1} &= \left(A\left(I + A^{-1}B\right)\right)^{-1} = \left(I + A^{-1}B\right)^{-1} A^{-1} \\
&= \left(I - A^{-1}B + \left(A^{-1}B\right)^2 - \left(A^{-1}B\right)^3 + \cdots\right) A^{-1} \\
&= A^{-1} - A^{-1}BA^{-1} + A^{-1}BA^{-1}BA^{-1} - \cdots .
\end{aligned}$$

Here, the third equality is based on the Neumann series [87] of $\left(I + A^{-1}B\right)^{-1}$. □

Next, we present the Jacobian matrices of $g^{AC}(u)$ and $g^{CH}(u)$ in Theorem (3.1.1).

Theorem 3.1.1. *The Jacobian matrix of $g^{AC}(u)$ (3.1) and $g^{CH}(u)$ (3.2) are*

$$J^{AC}(u) = A_1^{-1}\left(\left(s + \frac{\gamma}{\epsilon^2}\right) I - \frac{2\gamma}{\epsilon^2}\operatorname{diag}\left(uA_1^{-1}\left(s + \frac{\gamma}{\epsilon^2}\right) u\right)\right), \tag{3.4}$$

and

$$J^{CH}(u) = A_2^{-1}\left(sI - \frac{\gamma}{\epsilon^2}L_p + \frac{2\gamma}{\epsilon^2}L_p\operatorname{diag}\left(uA_2^{-1}\left(sI - \frac{\gamma}{\epsilon^2}L_p\right) u\right)\right), \tag{3.5}$$

where $A_1 = \left(-\gamma L_p + \frac{\gamma}{\epsilon^2}\operatorname{diag}(u^2) + sI\right)$ and $A_2 = \gamma L_p^2 - \frac{\gamma}{\epsilon^2}L_p\operatorname{diag}(u^2) + sI$.

Proof. Firstly, we derive the Jacobian matrix of $g^{AC}(u)$. Using lemma (3.1.1), we have

$$\begin{aligned}
g^{AC}(u+\delta u) &= \left(-\gamma L_p + \frac{\gamma}{\epsilon^2}\operatorname{diag}(u+\delta u)^2 + sI\right)^{-1}\left(s + \frac{\gamma}{\epsilon^2}\right)(u+\delta u) \\
&= \left(\underbrace{-\gamma L_p + \frac{\gamma}{\epsilon^2}\operatorname{diag}(u^2) + sI}_{A_1} + \underbrace{\frac{2\gamma}{\epsilon^2}\operatorname{diag}(u\delta u) + \mathcal{O}(\delta u^2)}_{B_1}\right)^{-1}\left(s + \frac{\gamma}{\epsilon^2}\right)(u+\delta u) \\
&= A_1^{-1}\left(s + \frac{\gamma}{\epsilon^2}\right)(u+\delta u) - A_1^{-1}\frac{2\gamma}{\epsilon^2}\operatorname{diag}(u\delta u)A_1^{-1}\left(s + \frac{\gamma}{\epsilon^2}\right)u + \mathcal{O}(\delta u^2)
\end{aligned}$$

It follows that

$$g^{AC}(u+\delta u)-g^{AC}(u)=A_1^{-1}\left(s+\frac{\gamma}{\epsilon^2}\right)\delta u-A_1^{-1}\frac{2\gamma}{\epsilon^2}\operatorname{diag}\left(uA_1^{-1}\left(s+\frac{\gamma}{\epsilon^2}\right)u\right)\delta u+\mathcal{O}(\delta u^2),$$

which leads to

$$J^{AC}(u)=A_1^{-1}\left(\left(s+\frac{\gamma}{\epsilon^2}\right)I-\frac{2\gamma}{\epsilon^2}\operatorname{diag}\left(uA_1^{-1}\left(s+\frac{\gamma}{\epsilon^2}\right)u\right)\right).$$

Similarly, we have

$$\begin{aligned}
&g^{CH}(u+\delta u)=\left(\gamma L_p^2-\frac{\gamma}{\epsilon^2}L_p\operatorname{diag}\left(u+\delta u\right)^2+sI\right)^{-1}\left(sI-\frac{\gamma}{\epsilon^2}L_p\right)(u+\delta u)\\
=&\left(\underbrace{\gamma L_p^2-\frac{\gamma}{\epsilon^2}L_p\operatorname{diag}(u^2)+sI}_{A_2}+\underbrace{\frac{2\gamma}{\epsilon^2}L_p\operatorname{diag}(u\delta u)+\mathcal{O}(\delta u^2)}_{B_2}\right)^{-1}\left(sI-\frac{\gamma}{\epsilon^2}L_p\right)(u+\delta u)\\
=&A_2^{-1}\left(sI-\frac{\gamma}{\epsilon^2}L_p\right)(u+\delta u)+A_2^{-1}\frac{2\gamma}{\epsilon^2}L_p\operatorname{diag}(u\delta u)A_2^{-1}\left(sI-\frac{\gamma}{\epsilon^2}L_p\right)u+\mathcal{O}(\delta u^2).
\end{aligned}$$

Then, we have

$$\begin{aligned}
&g^{CH}(u+\delta u)-g^{CH}(u)\\
=&A_2^{-1}\left(sI-\frac{\gamma}{\epsilon^2}L_p\right)\delta u+A_2^{-1}\frac{2\gamma}{\epsilon^2}L_p\operatorname{diag}\left(uA_2^{-1}\left(sI-\frac{\gamma}{\epsilon^2}L_p\right)u\right)\delta u+\mathcal{O}(\delta u^2),
\end{aligned}$$

and therefore

$$J^{CH}(u)=A_2^{-1}\left(sI-\frac{\gamma}{\epsilon^2}L_p+\frac{2\gamma}{\epsilon^2}L_p\operatorname{diag}\left(uA_2^{-1}\left(sI-\frac{\gamma}{\epsilon^2}L_p\right)u\right)\right).$$

□

For $g^{AC}(u)$, at the fixed points u_1^*, we have $A_1^{-1}(s+\frac{\gamma}{\epsilon^2})u_1^*=u_1^*$, and the Jacobian matrix becomes

$$J^{AC}(u_1^*)=\left(-\gamma L_p+\frac{\gamma}{\epsilon^2}\operatorname{diag}(u_1^{*2})+sI\right)^{-1}\left(\left(s+\frac{\gamma}{\epsilon^2}\right)I-\frac{2\gamma}{\epsilon^2}\operatorname{diag}\left(u_1^{*2}\right)\right).$$

Further suppose that $u_1^{*2} \approx \mathbf{1}$, then

$$J^{AC}(u_1^*) \approx \left(-\gamma L_p + \left(\frac{\gamma}{\epsilon^2} + s\right) I\right)^{-1} \left(s - \frac{\gamma}{\epsilon^2}\right) I.$$

Let $\lambda \geq 0$ be an eigenvalue of $-L_p$. It follows that the eigenvalues of above approximate Jacobian at u_1^* are

$$\frac{s - \frac{\gamma}{\epsilon^2}}{\gamma\lambda + \frac{\gamma}{\epsilon^2} + s}$$

Obviously, for any $s > 0$, we have $|s - \frac{\gamma}{\epsilon^2}| < s + \frac{\gamma}{\epsilon^2} \leq \gamma\lambda + \frac{\gamma}{\epsilon^2} + s$. This means that this Picard iteration (3.1) is strictly contractive at and near u_1^* if $u_1^{*2} \approx 1$. It also suggests that letting $s = \frac{\gamma}{\epsilon^2}$ may lead to very rapid convergence near such u_1^*. Nevertheless, such a rapid convergence may not be realized at u far from such u_1^*, and also not realized if u_1^{*2} is not very close to $\mathbf{1}$.

Similarly, for $g^{CH}(u)$, at the fixed points u_2^*, we know $A_2^{-1}\left(sI - \frac{\gamma}{\epsilon^2} L_p\right) u_2^* = u_2^*$, and assume that $u_2^{*2} \approx \mathbf{1}$, then

$$J^{CH}(u_2^*) \approx (\gamma L_p^2 - \frac{\gamma}{\epsilon^2} L_p + sI)^{-1} \left(sI + \frac{\gamma}{\epsilon^2} L_p\right).$$

Note that $(\gamma L_p^2 - \frac{\gamma}{\epsilon^2} L_p + sI)^{-1}$ and $sI + \frac{\gamma}{\epsilon^2} L_p$ have the same eigenvectors. Therefore, the eigenvalues of above approximate Jacobian at u_2^* are

$$\frac{s - \frac{\gamma}{\epsilon^2}\lambda}{\gamma\lambda^2 + \frac{\gamma}{\epsilon^2}\lambda + s}.$$

Let $0 = \lambda_1 < \lambda_2 < \cdots < \lambda_n$ be the eigenvalues of $-L_p$. Given that $s > 0$, for λ_1, the approximate $J^{CH}(u_2^*)$ has a corresponding eigenvalue 1, associate with the eigenvector $\mathbf{1}$. For $\lambda_2, \cdots, \lambda_n$, the corresponding eigenvalues of the approximate $J^{CH}(u_2^*)$ are strictly less than 1 in absolute value. In this sense, in the $n-1$ dimensional space orthogonal to $\mathbf{1}$, near u_2^* with $u_2^* \approx 1$, this Picard iteration (3.2) is strictly contractive. It is easy to check that the optimal s^* is given by $\frac{\gamma}{\epsilon^2}\lambda_2$ or $\frac{\gamma}{\epsilon^2}\lambda_n$. The only eigenvalue 1 also suggests that this Picard iteration does not change the component of $\mathbf{1}$ in u_k to the next iterate u_{k+1}. Overall, by the argument of continuity of eigenvalues of symmetric matrices that depend on the matrix elements, if u is sufficiently close to a vector whose elements are only $+1$ and -1, the Picard iterations (3.1) and (3.2) should still be strictly contractive near the fixed-point vector u^*. Note that, the tuning parameters s_1 and s_2 are better to be decided by

trial and error, although we provide the optimal choices theoretically when the methods approach convergence.

The contractive mappings guarantee the convergence of the fixed point iterations, which usually exhibit only linear convergence (unacceptably slow). Acceleration methods can potentially alleviate slow convergence. Here, we employ the Anderson acceleration [99] method to speed up the convergence. Anderson acceleration is a computational technique designed for accelerating the convergence of fixed point iterations [5,6], which is to find a proper linear combination of successive previous iterates with coefficients obtained from a constrained minimization to obtain a new iterate that potentially yields a smaller nonlinear residual norm than the new iterate computed from the original fixed point iteration.

A general framework of the Picard iterations with Anderson acceleration is provided in Algorithm 1. Here, the depth parameter m_k and relaxation factors c_k can be updated during the iterations to accelerate the convergence. Note that when $m_k = 0$, the iterations is exactly the original fixed point iteration. To further save computational cost, an inexact version of this algorithm can be used, which only requires to compute the $g(u_k)$ in a low accuracy. More details and analysis can be found in [104]. In our numerical experiments, we use the MATLAB built-in functions `pcg` and `gmres` to solve (3.1) and (3.2) to a relative accuracy, e.g., `lintol` $=10^{-3}$, respectively.

Algorithm 1 The Picard iterations with Anderson acceleration.

1: Start with an initial approximation u_0, Anderson acceleration depth $m_k \geq 0$ and relaxation factors $0 < c_k \leq 1$.
2: Compute $u_1 = g(u_0)$, and $w_1 = g(u_0) - x_0$.
3: **while** not converged **do**
4: Compute $g(u_k)$, and $w_k = g(u_k) - u_k$.
5: Let $\ell = \max\{0, k - m_k\}$, and solve $\min_{\sum_{i=\ell}^{k} \alpha_i^{(k)} = 1} \| \sum_{i=\ell}^{k} \alpha_i^{(k)} w_i \|$ for $\{\alpha_i^{(k)}\}$.
6: Compute the new iterate $u_{k+1} = (1 - c_k) \sum_{j=\ell}^{k} \alpha_j^{(k)} u_j + c_k \sum_{j=\ell}^{k} \alpha_j^{(k)} g(u_j)$.
7: $k = k + 1$
8: **end while**

3.2 Exponential time difference scheme

In this section, we present the fully discrete exponential time difference (ETD) scheme for Allen-Cahn (1.9) and Cahn-Hilliard (1.10) equations. Let the system of ODEs of spatially discretized AC/CH be $u_t = F(u)$. Also, let Lu be the linear part of $F(u)$ and $N(u)$ be the nonlinear part

of $F(u)$, respectively, such that $F(u) = Lu + N(u)$. Therefore, the Allen-Cahn equation can be written and decomposed as

$$u_t = \underbrace{\gamma\left(L_p - \frac{\kappa}{\epsilon^2}I\right)u}_{Lu} + \underbrace{\gamma(-1)\frac{f(u) - \kappa u}{\epsilon^2}}_{N(u)}, \tag{3.6}$$

and the Cahn Hilliard equation can be written and decomposed as

$$u_t = \underbrace{\gamma\left(-L_p^2 + \frac{\kappa}{\epsilon^2}L_p\right)u}_{Lu} + \underbrace{\gamma L_p\frac{f(u) - \kappa u}{\epsilon^2}}_{N(u)}, \tag{3.7}$$

where $\kappa \geq 2$ is a stabilizing parameter [37]. Given a positive integer K_t, the time interval is divided by $\{t_n = n\tau : 0 \leq n \leq K_t\}$ with a uniform time step $\tau = \frac{T}{K_t}$. The exponential Euler's method (ETD1) for AC/CH are as follows: for $k = 0, \cdots, K_t - 1$,

$$u_{k+1} = e^{\tau L}u_k + \tau\varphi_1(\tau L)N(u_k), \text{ where } \varphi_1(z) = \frac{e^z - 1}{z}. \tag{3.8}$$

The exponential trapezoid method (ETDRK2) for AC/CH takes the following form: for $k = 0, \cdots, K_t - 1$,

$$\hat{u}_{k+1} = e^{\tau L}u_k + \tau\varphi_1(\tau L)N(u_k), \tag{3.9a}$$

$$u_{k+1} = \hat{u}_{k+1} + \tau\varphi_2(\tau L)(N(\hat{u}_{k+1}) - N(u_k)), \text{ where } \varphi_2(z) = \frac{e^z - 1 - z}{z^2}. \tag{3.9b}$$

The above expressions show that a crucial computational component of the exponential integrator method is the exponential matrix-vector multiplication of the form $e^{\tau L}u$, where L is a symmetric negative definite matrix. An efficient method is provided in [80] for approximating $e^{\tau L}u = \varphi_0(\tau L)u$, $\varphi_1(\tau L)u$ and $\varphi_2(\tau L)u$, which only need solutions to 7 linear systems of the form $(\tau L - z_i I)x = c_i b$, where z_i and c_i are the coefficients of the rational approximation $e^z \approx r_\infty + \sum_{i=1}^{p}\frac{c_i}{z - z_i}$, where $p = 13$ or 14. Solving linear systems of the form $(L + \sigma I)x = b$ only requires one pair of FFT/IFFT in FPSM, because the coefficient matrix $L + \sigma I$ is a polynomial of the discrete Laplacian L_p. Also, e^z can be evaluated directly with only pair of FFT/IFFT in FPSM, which is adopted in our implementation. More importantly, using the ETD scheme can significantly increase the time step size, compared with the basic explicit methods.

In [37], a full analysis of the ETD1 and ETDRK2 for the AC equation is presented, where it is shown that the ETD1 and ETDRK2 schemes satisfy the discrete maximum principle, while, the discrete energy stability is proved for the ETD1 scheme. In the following of this section, we prove the discrete energy stability and 2-norm stability of the ETD1 scheme for CH equation. Remind that the discrete maximum principle does not hold true for the CH equation, unless for some special potential functions. The following Theorem 3.2.1 shows the discrete energy stability for CH ETD1 scheme (3.8).

Theorem 3.2.1. *Assume that* $\|u_k\|_\infty \leq 1$ *for* $k = 0, \cdots, K_t - 1$*, hence, the approximating solutions* $\{u_k\}_{k=0}^{K_t}$ *generated by ETD1 scheme* (3.8) *for Cahn-Hilliard equation satisfies the energy inequality*

$$E\left(u_{k+1}\right) \leq E\left(u_k\right), \quad 0 \leq k \leq K_t - 1,$$

for any $\tau > 0$*; i.e., the ETD1 scheme is unconditionally energy stable.*

Proof. Under the assumption $\|u_k\|_\infty \leq 1$ for $k = 0, \cdots, K_t - 1$, we have the difference between the discrete energies at two consecutive time levels from [37, Theorem 5.1] satisfying,

$$E\left(u_{k+1}\right) - E\left(u_k\right) \leq \left(u_{k+1} - u_k\right)^T \left(\left(-\epsilon^2 L_p + \kappa I\right) u_{k+1} - \left(\left(\kappa + 1\right) u_k - u_k^3\right)\right). \tag{3.10}$$

For the simplicity of notations, we focus on the following scaled equation, by multiplying both sides of (3.7) by $\frac{\epsilon^2}{\gamma}$. The new form reads

$$u_t = \underbrace{\left(-\epsilon^2 L_p^2 + \kappa L_p\right) u}_{Lu} + \underbrace{L_p\left(f\left(u\right) - \kappa u\right)}_{N(u)}. \tag{3.11}$$

Using (3.8), we have

$$\begin{aligned} u_{k+1} &= e^{\tau L} u_k + \tau \varphi_1\left(\tau L\right) N\left(u_k\right) \\ &= e^{\tau L} u_k + L^{-1}\left(e^{\tau L} - I\right) L_p\left(u_k^3 - \left(\kappa + 1\right) u_k\right). \end{aligned}$$

Then, we have

$$\begin{aligned}\left(u_k^3-(\kappa+1)\,u_k\right) &= L_p^{-1}\left(e^{\tau L}-I\right)^{-1} L\left(u_{k+1}-e^{\tau L}u_k\right)\\ &= \left(e^{\tau L}-I\right)^{-1}\left(-\epsilon^2 L_p+\kappa I\right)\left(u_{k+1}-e^{\tau L}u_k\right)\\ &= \left(e^{\tau L}-I\right)^{-1}\left(-\epsilon^2 L_p+\kappa I\right)\left(u_{k+1}-u_k\right)-\left(-\epsilon^2 L_p+\kappa I\right)u_k\end{aligned}$$

Note that L_p, L, $-\epsilon^2 L_p+\kappa I$, $e^{\tau L}-I$ and their (pseudo) inverses all have the same set of eigenvectors, and therefore their multiplications satisfy the commutative property. Also, the action of L_p^{-1} is well-defined for all vectors in the $n-1$-dimensional subspace orthogonal to $v_1=\mathbf{1}$.

Then, we have

$$\begin{aligned}&\left(-\epsilon^2 L_p+\kappa I\right)u_{k+1}-\left((\kappa+1)\,u_k-u_k^3\right)\\ =&\left(-\epsilon^2 L_p+\kappa I\right)\left(u_{k+1}-u_k\right)-\left(I-e^{\tau L}\right)^{-1}\left(-\epsilon^2 L_p+\kappa\right)\left(u_{k+1}-u_k\right)\\ =&B\left(u_{k+1}-u_k\right),\end{aligned}$$

where $B=\left(-\epsilon^2 L_p+\kappa I\right)-\left(I-e^{\tau L}\right)^{-1}\left(-\epsilon^2 L_p+\kappa I\right)$. Define a function

$$d(a):=a-\frac{a}{1-e^{-\frac{\tau a(a-\kappa)}{\epsilon^2}}},\quad a\geq\kappa.$$

Then, $d(a)<0$ for any $a\geq\kappa$ and $B=d\left(-\epsilon^2 L_p+\kappa I\right)$. Since $-\epsilon^2 L_p+\kappa I$ is symmetric and positive definite, we know that B is symmetric and negative definite. Therefore, we obtain

$$E\left(u_{k+1}\right)-E\left(u_k\right)\leq\left(u_{k+1}-u_k\right)^T B\left(u_{k+1}-u_k\right)\leq 0,$$

as desired. □

Note that Theorem 3.2.1 is based on the assumption $\|u\|_\infty\leq 1$, which is not satisfied by the solution for the CH equation. Therefore, we need to explore further to relax this assumption. Next, we prove the 2-norm stability for the CH equation in Theorem 3.2.2.

Theorem 3.2.2. *Assume that κ and τ are sufficiently large, hence, the approximating solutions*

$\{u_k\}_{k=0}^{K_t}$ *generated by ETD1 scheme* (3.8) *for Cahn-Hilliard equation satisfies*

$$\|u_{k+1}\|_2 \leq \|u_k\|_2, \quad 0 \leq k \leq K_t - 1.$$

Proof. Here, we still focus on the scaled equation (3.11) without changing any essence of the insight. Applying the ETD1 scheme, we have

$$\begin{aligned} u_{k+1} &= e^{\tau L} u_k + L^{-1} \left(e^{\tau L} - I\right) L_p \left(u_k^3 - (\kappa+1)\, u_k\right) \\ &= e^{\tau L} u_k + \left(-\epsilon^2 L_p + \kappa I\right)^{-1} \left(e^{\tau L} - I\right) \left(u_k^3 - (\kappa+1)\, u_k\right) \end{aligned} \tag{3.12}$$

With the periodic conditions, we know L_p is symmetric and negative definite. Assume that $\lambda_n \leq \cdots \leq \lambda_2 < \lambda_1 = 0$ are the eigenvalues of L_p and v. Therefore, we can let

$$L_p = V \operatorname{diag}(0, \lambda_2, \cdots, \lambda_n) V^T$$

and artificially define

$$L_p^{-1} = V \operatorname{diag}(0, \frac{1}{\lambda_2}, \cdots, \frac{1}{\lambda_n}) V^T,$$

which can be applied to any vector orthogonal to v_1. It follows that

$$\begin{aligned} L = -\epsilon^2 L_p^2 + \kappa L_p &= V \operatorname{diag}(0, -\epsilon^2 \lambda_2^2 + \kappa \lambda_2, \cdots, -\epsilon^2 \lambda_n^2 + \kappa \lambda_n) V^T \\ &= V \operatorname{diag}(0, \mu_2, \cdots, \mu_n) V^T, \end{aligned}$$

where $\mu_1 = 0$, $\mu_k = -\epsilon^2 \lambda_k^2 + \kappa \lambda_k < \kappa \lambda_k < 0$ $(2 \leq k \leq n)$, and

$$e^{\tau L} = V \operatorname{diag}(e^{\tau 0}, e^{\tau \mu_2}, \cdots, e^{\tau \mu_n}) V^T.$$

Let $u_{k+1} = Vw = \sum_{j=1}^n v_j w_j$ and $u_k^3 - (\kappa+1) u_k = Vr = \sum_{j=1}^n v_j r_j$ (subscript step k of w

and r are omitted for simplicity), and $V_2 = [v_2, \cdots, v_n]$. Plugging them into (3.12), we have

$$\begin{aligned} u_{k+1} &= V[w_1, e^{\tau\mu_2} w_2, \cdots, e^{\tau\mu_n} w_n]^T + V[0, \frac{e^{\tau\mu_2}-1}{\epsilon^2|\lambda_2|+\kappa} r_2, \cdots, \frac{e^{\tau\mu_n}-1}{\epsilon^2|\lambda_n|+\kappa} r_n]^T \\ &= v_1 w_1 + V_2[e^{\tau\mu_2} w_2, \cdots, e^{\tau\mu_n} w_n]^T + V_2[\frac{e^{\tau\mu_2}-1}{\epsilon^2|\lambda_2|+\kappa} r_2, \cdots, \frac{e^{\tau\mu_n}-1}{\epsilon^2|\lambda_n|+\kappa} r_n]^T \\ &= v_1 w_1 + V_2[e^{\tau\mu_2}, \cdots, e^{\tau\mu_n}]^T.[w_2, \cdots, w_n]^T \\ &\qquad + V_2[\frac{e^{\tau\mu_2}-1}{\epsilon^2|\lambda_2|+\kappa}, \cdots, \frac{e^{\tau\mu_n}-1}{\epsilon^2|\lambda_n|+\kappa}]^T.[r_2, \cdots, r_n]^T, \end{aligned} \tag{3.13}$$

where the "." operation means element-wise multiplication between two vectors. Applying the orthogonal projector $P_2 = I - \frac{v_1 v_1^T}{v_1^T v_1}$ on both sides, we have

$$P_2 u_{k+1} = V_2[e^{\tau\mu_2}, \cdots, e^{\tau\mu_n}]^T.[w_2, \cdots, w_n]^T + V_2[\frac{e^{\tau\mu_2}-1}{\epsilon^2|\lambda_2|+\kappa}, \cdots, \frac{e^{\tau\mu_n}-1}{\epsilon^2|\lambda_n|+\kappa}]^T.[r_2, \cdots, r_n]^T, \tag{3.14}$$

To obtain an upper bound on $\|P_2 u_{k+1}\|_2$, we can apply a variant of Hölder's inequality $\|u.v\| \leq \|u\|_p \|v\|_q$, where $\frac{1}{p} + \frac{1}{q} = \frac{1}{r}$. Let $p = \infty$, $q = 2$, and $r = 2$. It follows that

$$\begin{aligned} &\|[e^{\tau\mu_2}, \cdots, e^{\tau\mu_n}].[w_2, \cdots, w_n]\|_2 \leq e^{\tau\mu_2} \|[w_1, \cdots, w_n]\|_2 \\ &= e^{\tau\mu_2} \|V2[w_2, \cdots, w_n]^T\|_2 = e^{\tau\mu_2} \|P_2 u_k\|_2 \end{aligned} \tag{3.15}$$

Similarly, the second term can be bounded by

$$\begin{aligned} &\left\| \left[\frac{e^{\tau\mu_2}-1}{\epsilon^2|\lambda_2|+\kappa}, \cdots, \frac{e^{\tau\mu_n}-1}{\epsilon^2|\lambda_n|+\kappa} \right].[r_2, \cdots, r_n] \right\|_2 \leq \max_{2\leq i\leq n} \left| \frac{1-e^{\tau\mu_i}}{\epsilon^2|\lambda_i|+\kappa} \right| \|[r_2, \cdots, r_n]\|_2 \\ &\leq \frac{1-e^{\tau\mu_\ell}}{\epsilon^2|\lambda_\ell|+\kappa} \|V_2[r_2, \cdots, r_n]\|_2 = \frac{1-e^{\tau\mu_\ell}}{\epsilon^2|\lambda_\ell|+\kappa} \left\| P_2(u_k^3 - (\kappa+1)u_k) \right\|_2 . \end{aligned} \tag{3.16}$$

Suppose that $\|u_K\|_\infty \leq U$, where $U \in (0, \sqrt{\kappa+1}]$ is a fixed constant independent of κ, and $P_2 u_k \neq 0$. We want to show that

$$\|P_2(u_k^3 - (\kappa+1)u_k)\|_2 \leq (1+\delta(\kappa)) \|(P_2)u_k^3 - (\kappa+1)P_2 u_k\|_2,$$

for a sufficiently large κ, where $\delta(\kappa) = \frac{\|P_2 u_k^3\|_2 + \|(P_2 u_k)^3\|_2}{\|(\kappa+1)P_2 u_k\|_2 - \|(P_2 u_k)^3\|_2}$, so that $\lim_{\kappa\to\infty} \delta(\kappa) = 0$. In fact, since $\|u_K\|_\infty \leq U$ and U is fixed independent of κ, $\|P_2 u_k^3\|_2$ and $\|(P_2 u_k)^3\|_2$ are bounded independent

of κ. It follows that

$$\begin{aligned}\|P_2(u_k^3-(\kappa+1)u_k)\|_2 &\le \|(\kappa+1)P_2u_k\|_2\left(1+\frac{\|P_2u_k^3\|_2}{\|(\kappa+1)P_2u_k\|_2}\right) \\ &= \|(\kappa+1)P_2u_k\|_2\left(1-\frac{\|P_2u_k^3\|_2}{\|(\kappa+1)P_2u_k\|_2}\right)(1+\delta(\kappa)) \\ &\le \|(P_2)u_k^3-(\kappa+1)P_2u_k\|_2(1+\delta(\kappa)).\end{aligned} \tag{3.17}$$

We need an upper bound on $\|(P_2)u_k^3-(\kappa+1)P_2u_k\|_2$. Let the m-th element of P_2u_k be a_m, and suppose that $\|P_2u_k\|_\infty = \max_m|a_m| \le \sqrt{\kappa+1}$. Mote that $P_2u_k = u_k - \bar{u}_k$ such that $\|P_2u_k\|_\infty \le \|u_k\|_\infty + |\bar{u}_k|$, and similarly, $\|u_k\|_\infty \le \|P_2u_k\|_\infty + |\bar{u}_k|$, where $\bar{u}_k$ is the mean value of the vector u_k. Under this assumption, note that larger a_k^2 corresponds to smaller $(\kappa+1)-a_k^2$ (both terms are non-negative). By the Chebyshev sum inequality [53], we have

$$\begin{aligned}&\|(P_2u_k)^3-(\kappa+1)P_2u_k\|_2^2 = \sum_{m=1}^n a_m^2((\kappa+1)-a_m^2)^2 \\ \le&(\sum_{m=1}^n a_m^2)\left(\frac{1}{n}\sum_{m=1}^n((\kappa+1)-a_k^2)^2\right) \\ =&\|P_2u_k\|_2^2\left((\kappa+1)^2-\frac{1}{n}\sum_{m=1}^n a_m^2(2(\kappa+1)-a_m^2)\right) \text{ (where } \kappa+1\le 2(\kappa+1)-a_m^2) \\ \le&\|P_2u_k\|_2^2\left((\kappa+1)^2-\frac{\kappa+1}{n}\sum_{m=1}^n a_m^2\right) \le \|P_2u_k\|_2^2\left((\kappa+1)-\frac{1}{2n}\|P_2u_k\|_2^2\right).\end{aligned} \tag{3.18}$$

Combined (3.14) with (3.15), (3.16), (3.17) and (3.18), we have

$$\begin{aligned}\|P_2u_{k+1}\|_2 &\le e^{\tau\mu_2}\|P_2u_k\|_2 + \frac{1-e^{\tau\mu_\ell}}{\epsilon^2|\lambda_\ell|+\kappa}\|P_2(u_k^3-(\kappa+1)u_k)\|_2 \\ &\le e^{\tau\mu_2}\|P_2u_k\|_2 + (1-e^{\tau\mu_\ell})\frac{(1+\delta(\kappa))\left((\kappa+1)-\frac{1}{2n}\|P_2u_k\|_2^2\right)}{\epsilon^2|\lambda_\ell|+\kappa}\|P_2u_k\|_2.\end{aligned} \tag{3.19}$$

Suppose that κ and the step size τ are sufficiently large, so that $\delta(\kappa)>0$ and $e^{\tau\mu_2}>0$ are sufficiently small. Then, if $\frac{1}{2n}\|P_2u_k\|_2 \le 1$ and $\epsilon > \sqrt{\frac{1-\frac{1}{2n}\|P_2u_k\|_2^2}{|\lambda_\ell|}}$, then

$$\|P_2u_{k+1}\|_2 \le \left(e^{\tau\mu_2}+(1-e^{\tau\mu_\ell})\frac{(1+\delta(\kappa))\left((\kappa+1)-\frac{1}{2n}\|P_2u_k\|_2^2\right)}{\epsilon^2|\lambda_\ell|+\kappa}\right)\|P_2u_k\|_2 \le \|P_2u_k\|_2.$$

Note that if $\frac{1}{2n}\|P_2u_k\|_2 > 1$, then $\|P_2u_{k+1}\|_2 \le \|P_2u_k\|_2$ holds for any ϵ as long as κ and τ are

sufficiently large. Finally, recall (3.13) showing that the v_1 component in the u_{k+1} is the same as that u_k, i.e., $\|P_1 u_{k+1}\|_2 = \|P_1 u_k\|_2$, where $P_1 = I - P_2 = \frac{v_1 v_1^T}{v_1^T v_1}$, because

$$u_k = VV^T u_k = v_1(v_1^T u_k) + V_2(V_2^T u_k) = P_1 u_k + P_2 u_k.$$

It follows that

$$\|u_{k+1}\|_2^2 = \|P_1 u_{k+1}\|_2^2 + \|P_2 u_{k+1}\|_2^2 \leq \|P_1 u_k\|_2^2 + \|P_2 u_k\|_2^2 = \|u_k\|_2^2,$$

which complete the proof. □

3.3 The preconditioned conjugate gradient method

Another alternative for computing the steady-state solutions of the AC and CH equations arises from an optimization perspective. In this section, we develop the preconditioned conjugate gradient method for computing the steady-state solutions of the AC and CH equations, which is equivalent to minimize the discretized energy functional (3.20). Let L_p be the discrete Laplacian matrix. In discrete form, the energy functional (1.13) can be written as

$$E(u) = \left(-\frac{1}{2} u^T L_p u + \frac{1}{4\epsilon^2} \left(u^T \mathrm{diag}\left(u^2\right) u - 2u^T u + \mathbf{1}^T \mathbf{1} \right) \right) h^d. \tag{3.20}$$

Then, our aim is to compute the local minimizer of the discrete energy (3.20). It is necessary to obtain the gradient of (3.20) in order to employ the PCG method. Note that, although the AC and CH equations have the same energy functional, the gradients are not essentially the same due to the mass conservative for CH equation. For the AC equation, the *effective* gradient of $E(u)$ with respect to u is

$$r^{AC}(u) = \frac{\partial E(u)}{\partial u} = -L_p u + \frac{1}{\epsilon^2}\left(u^3 - u\right), \tag{3.21}$$

For the CH equation, in order to reflect the direction of change with respect to time specified by the right-hand side of (1.10), a *modified* gradient to preserve the mass is

$$r^{CH}(u) = -L_p\left(-L_p u + \frac{1}{\epsilon^2}\left(u^3 - u\right)\right). \tag{3.22}$$

Here, we disregard the the factors h^d for both AC and CH equations. With the gradient expressions (3.21) and (3.22), a nonlinear preconditioned conjugate gradient (PCG) method could be employed in an effort to find the steady states of the energy functional (3.20) for AC and CH equations. Suppose we work with a generic preconditioner M. The standard search direction in nonlinear PCG is

$$p_k = -M^{-1}r_k + \beta_k p_{k-1}, \tag{3.23}$$

with $p_0 = -r_0$ and the Polak–Ribière update [76]

$$\beta_k = \max\left(\frac{\langle r_k - r_{k-1}, M^{-1}r_k\rangle}{\langle r_{k-1}, M^{-1}r_{k-1}\rangle}, 0\right). \tag{3.24}$$

At iteration k, a regular update formula for u_{k+1} in PCG could be

$$u_{k+1} = u_k + \eta_k p_k, \tag{3.25}$$

where η_k is the step size, which can be determined by various methods, e.g. exact line search or a backtracking line search with Armijo–Goldstein condition [13]. Generally, the exact line search is prohibitive for large problems due to the requirement of evaluations of the energy functional at many points, which leads to extremely high computational cost. Here, we provide an efficient way to evaluate the energies at different points.

Let u_k be the current approximate solution, and p_k be the search direction. We want to determine the optimal step size η_k such that $u_{k+1} = u_k + \eta_k p_k$ achieves the minimum energy along p_k. We can write $u_{k+1} = \begin{pmatrix} u_k & p_k \end{pmatrix}\begin{pmatrix} 1 & \eta \end{pmatrix}^T$ and plug into $E(u)$ (3.20), and the energy becomes

a function of the single variable η. We obtain

$$E(u_k+\eta p_k)=\Bigg[-\frac{1}{2}\begin{pmatrix}1 & \eta\end{pmatrix}\left(\begin{pmatrix}u_k & p_k\end{pmatrix}^T L_p\begin{pmatrix}u_k & p_k\end{pmatrix}\right)\begin{pmatrix}1 & \eta\end{pmatrix}^T \tag{3.26}$$
$$+\frac{1}{4\epsilon^2}\left(\begin{pmatrix}1 & \eta & \eta^2\end{pmatrix}\begin{pmatrix}u_k^2 & 2u_kp_k & p_k^2\end{pmatrix}^T\begin{pmatrix}u_k^2 & 2u_kp_k & p_k^2\end{pmatrix}\begin{pmatrix}1 & \eta & \eta^2\end{pmatrix}^T\right.$$
$$\left.-2\begin{pmatrix}1 & \eta\end{pmatrix}\begin{pmatrix}u_k & p_k\end{pmatrix}^T\begin{pmatrix}u_k & p_k\end{pmatrix}\begin{pmatrix}1 & \eta\end{pmatrix}^T+n\right)\Bigg]h^d.$$

Here, u_kp_k stands for the column vector whose entries are the product of those of u_k and p_k; u_k^2 and p_k^2 can be defined similarly.

Therefore, evaluation of $E(u_k+\eta p_k)$ at different candidate values of η requires the computation of the 2×2 matrices $\begin{pmatrix}u_k & p_k\end{pmatrix}^T L_p\begin{pmatrix}u_k & p_k\end{pmatrix}$ and $\begin{pmatrix}u_k & p_k\end{pmatrix}^T\begin{pmatrix}u_k & p_k\end{pmatrix}$, and the 3×3 matrix $\begin{pmatrix}u_k^2 & 2u_kp_k & p_k^2\end{pmatrix}^T\begin{pmatrix}u_k^2 & 2u_kp_k & p_k^2\end{pmatrix}$ only once and *no more* computation in the original problem dimension n is needed. Now $E(u_k+\eta p_k):\mathbb{R}\to\mathbb{R}$ can be evaluated for any and as many values of η as needed at little arithmetic cost. We can afford to perform a numerical exact line search to minimize $E(u_k+\eta p_k)$, or find the minimizer by forming $\frac{d}{d\eta}E(u_k+\eta p_k)=0$ in closed form and solving it for η. In our implementation, we use MATLAB's `fminsearch` function to find the optimal η, which can be done rapidly without additional work on dimension n. Note that the fast evaluation of the energy functional $E(u)$ is essential to enable the exact line search in the PCG method. Also, the fast evaluation of the energy is still advantageous for the other line search methods, which require to evaluate the energy functional at more than one point.

Another critical problem in the nonlinear PCG method is to design a good preconditioner, which can significantly reduce the iteration counts and runtime. In general, the preconditioner for PCG near convergence should be an approximation to the Hessian of the objective function. For the AC equation, it is easy to get the *effective* Hessian of $E(u)$, which is

$$H^{AC}(u)=\frac{\partial r_1(u)}{\partial u}=-L_p+\frac{1}{\epsilon^2}\left(3\mathrm{diag}\left(u^2\right)-I\right). \tag{3.27}$$

Similarly, for the CH equation, it is natural to use the Jacobian matrix of $r_2(u)$ (3.22) as the

preconditioner, which is

$$H^{CH}(u) = \frac{\partial r_2(u)}{\partial u} = L_p^2 - \frac{1}{\epsilon^2} L_p \left(3\text{diag}\left(u^2\right) - I\right). \tag{3.28}$$

Note that, although, the preconditioning could accelerate the convergence of the PCG method, we still need to consider the computational cost of applying the preconditioners. A preconditioner, which can be applied efficiently is crucial for the success of the PCG method. Given (3.27) and (3.28), applying the preconditioners in the PCG method require some iterative methods to solve the linear systems $M^{-1}r_k$ involved in (3.23). To lower the computational cost, the geometric multigrid (GMG) method might be a reasonable method to approximate the action of M^{-1} on vectors. However, these preconditioning strategies can be still expensive, especially for the 3D problems. Usually, the action of preconditioning does not need to be computed to high accuracy. Also, we know that the elements of the steady state solutions AC and CH equations are around 1 or -1. Therefore, we introduce two efficient preconditioners for AC and CH in the PCG method, which are

$$M^{AC} = -L_p + \frac{2}{\epsilon^2} I, \quad \text{and} \quad M^{CH} = L_p^2 - \frac{2}{\epsilon^2} L_p, \tag{3.29}$$

by assuming that $u^2 = \mathbf{1}$ in (3.27) and (3.28), respectively. Since we adopt the Fourier pseudo-spectral discretization scheme, applying the above preconditioners in the PCG method only require one pair of FFT/IFFT, respectively, as both preconditioners are polynomials of the discrete Laplacian L_p.

With the preconditioners(3.29) applied on the gradient (3.21) or (3.22), the convergence of the nonlinear PCG relies on the properties of the preconditioned Hessian operator [1, 79]. Then, it is reasonable to expect that the convergence of the nonlinear PCG method will be influenced by the properties of the operators

$$(M^{AC})^{-\frac{1}{2}} H^{AC} (M^{AC})^{-\frac{1}{2}} \tag{3.30}$$

for the AC equation and

$$(M^{CH})^{-\frac{1}{2}} H^{CH} (M^{CH})^{-\frac{1}{2}} \tag{3.31}$$

for the CH equation, respectively. Next, we show the pattern of the spectrum for the operators (3.30) and (3.31), such that the nonlinear PCG is expected to converge quickly when approaching convergence.

Note that the elements of the steady state solutions AC and CH equations are around 1 or -1. For the steady-state solution u^* of the AC and CH equation, we can assume that $u^{*2}+\delta u^{*2} = \mathbf{1}$, where $\delta u^{*2} \in \mathbb{R}^n$. Then, for the AC equation, most of the elements of δu^{*2} are exactly 0, and for the CH equation, most of the elements of δu^{*2} are nearly 0. Also, it is well-known that solutions of the AC and CH equation will develop interfaces with thickness $\mathcal{O}(\epsilon)$ [28, 60], which render theirs numerical simulation difficult as resolving thin interfaces. Thus, the ratio between the number of the elements of u^* which are not close to 1 or -1 and the number of elements in u^* that are (very close to) 1 or -1 is proportional to ϵ, i.e, $\mathcal{O}(\epsilon)$. The following Theorems 3.3.1 and 3.3.2 clearly shows that the operator (3.30) or (3.31) has a favorable eigenvalue distribution such that the nonlinear PCG with our proposed preconditioners(3.29) is expected to converge fairly quickly when approaching convergence.

Theorem 3.3.1. *At the steady-state solutions u^* of the AC equation, $n(1-\mathcal{O}(\epsilon))$ eigenvalues of the operator* (3.30) *are exactly* 1*, while $n\mathcal{O}(\epsilon)$ eigenvalues are significantly different from 1.*

Proof. To show the spectrum of the operator $(M^{AC})^{-\frac{1}{2}} H^{AC} (M^{AC})^{-\frac{1}{2}}$, it is equivalent to consider the operator $(M^{AC})^{-1} H^{AC}$, since they have the same eigenvalues. For the AC equation, it is easy to get

$$\begin{aligned} H^{AC} &= -L_p + \frac{1}{\epsilon^2}\left(3\mathrm{diag}\left(u^{*2}\right) - I\right) \\ &= -L_p + \frac{1}{\epsilon^2}\left(3\mathrm{diag}\left(u^{*2}\right) + 3\,\mathrm{diag}\left(\delta u^{*2}\right) - I\right) - \frac{3}{\epsilon^2}\,\mathrm{diag}(\delta u^{*2}) \\ &= -L_p + \frac{2}{\epsilon^2} I - \frac{3}{\epsilon^2}\,\mathrm{diag}(\delta u^{*2}) \end{aligned}$$

Also, for the AC equation, we know that $n\left(1-\mathcal{O}(\epsilon)\right)$ elements of δu^{*2} are exactly 0, while only $n\mathcal{O}(\epsilon)$ elements are not 0. Then, with the singular value decomposition, we can write the low-rank matrix $\frac{3}{\epsilon^2}\,\mathrm{diag}(\delta u^{*2}) = U_1 C_1 V_1^T$, where $U_1, V_1 \in \mathbb{R}^{n \times n\mathcal{O}(\epsilon)}$ and $C_1 \in \mathbb{R}^{n\mathcal{O}(\epsilon) \times n\mathcal{O}(\epsilon)}$. Based on the

Sherman–Morrison–Woodbury formula [49, 54, 85], we have

$$\begin{aligned}(M^{AC})^{-1}H^{AC} &= (H^{AC} + U_1 C_1 V_1^T)^{-1} H^{AC} \\ &= I - (H^{AC})^{-1} U_1 \left(C_1^{-1} + V_1^T (H^{AC})^{-1} U_1\right)^{-1} V_1^T,\end{aligned}$$

which is a rank-$n\mathcal{O}(\epsilon)$ update of the identity matrix I. This implies that $n(1-\mathcal{O}(\epsilon))$ eigenvalues of $(M^{AC})^{-1}H^{AC}$ are exactly 1 and there are only $n\mathcal{O}(\epsilon)$ eigenvalues that could be significantly different from 1. □

Theorem 3.3.2. *At the steady-state solutions u^* of the CH equation, $n\,(1-\mathcal{O}(\epsilon))$ eigenvalues of the operators* (3.30) *are nearly* 1*, while $n\mathcal{O}(\epsilon)$ eigenvalues are significantly different from 1.*

Proof. Similarly, for the CH equation, we consider the spectrum of the operator $\left(M^{CH}\right)^{-1} H^{CH}$. We have

$$\begin{aligned}H^{CH} &= L_p^2 - \frac{1}{\epsilon^2} L_p \left(3\mathrm{diag}\left(u^{*2}\right) - I\right). \\ &= L_p^2 - \frac{1}{\epsilon^2} L_p \left(3\mathrm{diag}\left(u^{*2}\right) + 3\,\mathrm{diag}(\delta u^{*2}) - I\right) + \frac{3}{\epsilon^2} L_p\,\mathrm{diag}(\delta u^{*2}) \\ &= L_p^2 - \frac{2}{\epsilon^2} L_p + \frac{3}{\epsilon^2} L_p\,\mathrm{diag}(\delta u^{*2})\end{aligned}$$

For the CH equation, we know that $n(1-\mathcal{O}(\epsilon))$ elements of δ are nearly 0, while $\mathcal{O}(\epsilon)$ elements are significantly different from 0. Then, we can write the full-rank matrix $\frac{3}{\epsilon^2} L_p\,\mathrm{diag}(\delta u^{*2}) = \frac{3}{\epsilon^2} L_p \left(\mathrm{diag}(\iota) + U_2 C_2 V_2^T\right)$. Here, $U_2, V_2 \in \mathbb{R}^{n \times n\mathcal{O}(\epsilon)}$, $C_2 \in \mathbb{R}^{n\mathcal{O}(\epsilon) \times n\mathcal{O}(\epsilon)}$, and $\iota \in \mathbb{R}^n$ such that $\|\iota\|_\infty \leq |\delta|$, where $\delta > 0$ is a small scalar.

Again, with the Sherman–Morrison–Woodbury formula, we have

$$\begin{aligned}\left(M^{CH}\right)^{-1} H^{CH} &= \left(H^{CH} - \frac{3}{\epsilon^2} L_p\,\mathrm{diag}(\iota) - \frac{3}{\epsilon^2} L_p U_2 C_2 V_2^T\right)^{-1} H^{CH} \\ &= \left(H^{CH} - \frac{3}{\epsilon^2} L_p\,\mathrm{diag}(\iota)\right)^{-1} H^{CH} + \left(H^{CH} - \frac{3}{\epsilon^2} L_p\,\mathrm{diag}(\iota)\right)^{-1} \\ &\quad U_2 \left(C_2^{-1} - V_2^T \left(H^{CH} - \frac{3}{\epsilon^2} L_p\,\mathrm{diag}(\iota)\right)^{-1} U_2\right)^{-1} V_2^T\end{aligned}$$

With Lemma 3.1.1, we know

$$\left(H^{CH} - \frac{3}{\epsilon^2} L_p \operatorname{diag}(\iota)\right)^{-1} H^{CH} = I + \frac{3}{\epsilon^2} (H^{CH})^{-1} L_p \operatorname{diag}(\iota) + \mathcal{O}(\|\iota\|_\infty^2).$$

Therefore,

$$\begin{aligned}\left(M^{CH}\right)^{-1} H^{CH} = I + \frac{3}{\epsilon^2} (H^{CH})^{-1} L_p \operatorname{diag}(\iota) + \mathcal{O}(\|\iota\|_\infty^2) + \left(H^{CH} - \frac{3}{\epsilon^2} L_p \operatorname{diag}(\iota)\right)^{-1} \\ U_2 \left(C_2^{-1} - V_2^T \left(H^{CH} - \frac{3}{\epsilon^2} L_p \operatorname{diag}(\iota)\right)^{-1} U_2\right)^{-1} V_2^T,\end{aligned}$$

which is a rank-$n\mathcal{O}(\epsilon)$ update of the identity matrix I plus a matrix of small norm (bounded by $\|\iota\|_\infty < \delta$). This implies that $n(1 - \mathcal{O}(\epsilon))$ eigenvalues of $(M^{CH})^{-1} H^{CH}$ are nearly 1 and there are only $n\mathcal{O}(\epsilon)$ eigenvalues that could be significantly different from 1. □

An outline of the nonlinear PCG is given in Algorithm 2.

Algorithm 2 The preconditioned conjugate gradient method.

1: Start with an initial approximation u_0, $k = 0$ and define $p_{-1} = \mathbf{0}$.
2: **while** not converged **do**
3: $r_k = r(u_k)$ (see (3.21) and (3.22)).
4: $\beta_k = \max\left(\frac{\langle r_k - r_{k-1}, M^{-1} r_k \rangle}{\langle r_{k-1}, M^{-1} r_{k-1} \rangle}, 0\right)$
5: $p_k = -M^{-1} r_k + \beta_k p_{k-1}$
6: $\eta_k = \arg\min_\eta E(u_k + \eta p_k)$ (by fast energy evaluation (3.26))
7: $u_{k+1} = u_k + \eta_k p_k$
8: $k = k + 1$
9: **end while**

3.4 Numerical experiments

In this section, we perform numerical experiments in 2D and 3D domains to show the performances of the three methods we introduced, i.e., the PCG method, the Picard iterative method with inexact AA, and the ETDRK2 method with large time steps. The time-scaling parameters [84] for the AC and CH equations are $\gamma = \epsilon^2$ and $\gamma = \epsilon^4$, respectively. In each experiment, the mesh size h is set to be $\epsilon/2$. For computing the steady-state solutions, we can define the relative change

of discrete energy $E(u)$ (3.20) as

$$E_k^{rel} = \frac{|E(u_{k+1}) - E(u_k)|}{\max\{1, |E(u_{(k)})|\}}. \tag{3.32}$$

Then, we adopt the stopping criterion, i.e., $E_k^{rel} \leq 64\epsilon_{mac}$. Here, ϵ_{mac} is the *machine precision*. Note that among these three methods, the PCG method is parameter-free. There is no other specified details for the PCG method. For the Picard iterations (3.1) and (3.2), the tuning parameters $s = 10^{-3}$ and 10^{-4} are selected for the AC and CH equations, respectively. For the Picard iterative method with inexact AA in Algorithm 1, the relaxation factors $c_k = 0.5$ is fixed for $k = 1, 2, \cdots$, and the depth parameters m_k is given in Table (3.1), where we adopt the multilevel depth strategy of small m_k early and large m_k late in the iteration [77]. At each step k, we only solve the equations (3.1)

Table 3.1: The values of m_k

E_k^{rel}	m_k
$[10^{-3}, \infty)$	0
$[10^{-4}, 10^{-3})$	2
$[10^{-5}, 10^{-4})$	4
$[10^{-6}, 10^{-5})$	5
$[0, 10^{-6})$	6

and (3.2) to a relative accuracy $\min\{10^{-3}, \sqrt{E_k^{rel}/10}\}$. For the ETDRK2 method, we specify the time step size $\tau = 10$. All the experiments in this section are performed on a single node with 16 cores on Clemson Palmetto Cluster running MATLAB R2022a.

3.4.1 Numerical experiments in 2D

In this section, we perform extensive experiments in 2D domains to compare the performances among the three methods. The computational domain is $\mathcal{D} = [-1, 1]^2$. To compare the performances among these three methods, we expect that the final solutions obtained by these methods are the same. To be fair enough, we first run the ETDRK2 ($\tau = 0.01$) with initial approximation with random entries uniformly distributed on $[-0.5, 0.5]$ and $[0, 1]$ for AC and CH equations, generated by MATLAB's `rand` function with the fixed seed 1. We save the approximation u_k generated by the ETDRK2 when $E_k^{rel} \leq 10^{-4}$. Then, we use this u_k as the initial approximation u_0 for all these three methods.

3.4.1.1 The AC equation

In this example, we perform several experiments to compare the performances among the three methods for the AC equation. The results are provided in Table 3.2. Here, "N" records the size the of the problem, "Iter" counts the total number of iterations, and "Time" is the runtime in seconds. Since all these methods converge to the same final solution u_g, we only report the corresponding discrete energy $E(u_g)$ in the last column "Energy". For example, in Table 3.2, when $\epsilon = \frac{1}{64}$ ($h = \frac{1}{128}$), it takes the PCG method 351 iterations in 3.17 seconds to obtain the final solution $u_g \in \mathbb{R}^{65536}$ with the discrete energy 241.35911438, whereas, the AA and ETDRK2 methods require 835 iterations in 21.51 and 6291 iterations in 40.25 seconds, respectively. Note that, if $\epsilon = \frac{1}{128}$ or $\frac{1}{512}$, all these three methods reach the trivial solution $-\mathbf{1}$. Figure 3.1 shows the contour plots of the density function u_g obtained with the PCG method, which implies that most of the elements of u_g are 1 or -1. We can see that the PCG is much more efficient than the other two methods. The performances of AA and ETDRK2 are competitive. When ϵ becomes smaller, it becomes more difficult for all the three methods, however, the PCG method gains more advantage over the other two methods. The PCG method achieves the fastest convergence in terms of iterations and is much cheaper in computational cost.

Table 3.2: Performance comparison for the 2D AC equation.

ϵ	N	PCG		AA		ETDRK2		Energy
		Iter	Time	Iter	Time	Iter	Time	
1/64	65536	351	3.17	835	21.51	6291	40.25	241.35911438
1/128	262144	973	27.21	6982	516.63	6056	98.07	0
1/256	1048576	1879	191.13	15737	5185.06	101546	5953.41	965.43645980
1/512	4194304	4966	2282.52	37802	47077.59	97183	31739.25	0

3.4.1.2 The CH equation

We compare the performances among the three methods for the CH equation in this example. We summarize the results in Table 3.3. Since the CH equation has the mass conservative property, with our initial approximation, it could avoid the trivial solution in this situation. We can see that computing the steady-state solution for the CH equation is more challenging than that for the AC equation, even for the PCG method. Moreover, we find that without the preconditioner, the PCG method could converge very slowly for the CH equation. This is reasonable, sine we use the *modified*

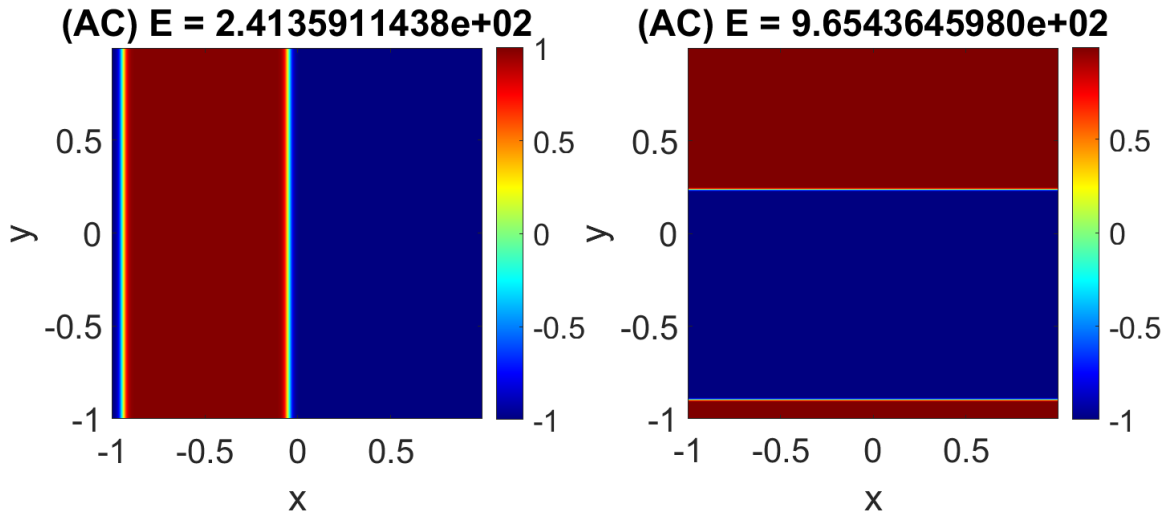


Figure 3.1: Corresponding contour plots of the steady-state solutions u_g for the 2D AC equation obtained by the PCG method (left : $\epsilon = \frac{1}{64}$, right: $\epsilon = \frac{1}{256}$).

gradient (3.22) for the CH equation, which reflects the direction of change with time specified by the continuous equation (1.10), but does not achieve the steepest descent direction with respect to the discrete energy (3.20). However, the PCG method still achieves the best performances in terms of the runtime among all these three methods. When ϵ becomes smaller, it becomes more challenging for all these methods and it seems not suitable to use the AA and ETDRK2 method to compute the steady-state solutions for the CH equation.

Table 3.3: Performance comparison for the 2D CH equation.

ϵ	N	PCG		AA		ETDRK2		Energy
		Iter	Time	Iter	Time	Iter	Time	
1/64	65536	1721	18.32	984	55.36	21947	257.49	213.19874621
1/128	262144	2336	85.51	5010	1014.83	188220	5618.84	427.17439097
1/256	1048576	7105	879.77	37637	24352.94	1016780	100684.09	854.90965010
1/512	4194304	34535	20719.82	63969+	181091.50+	1038816+	606573.89+	1710.6116834

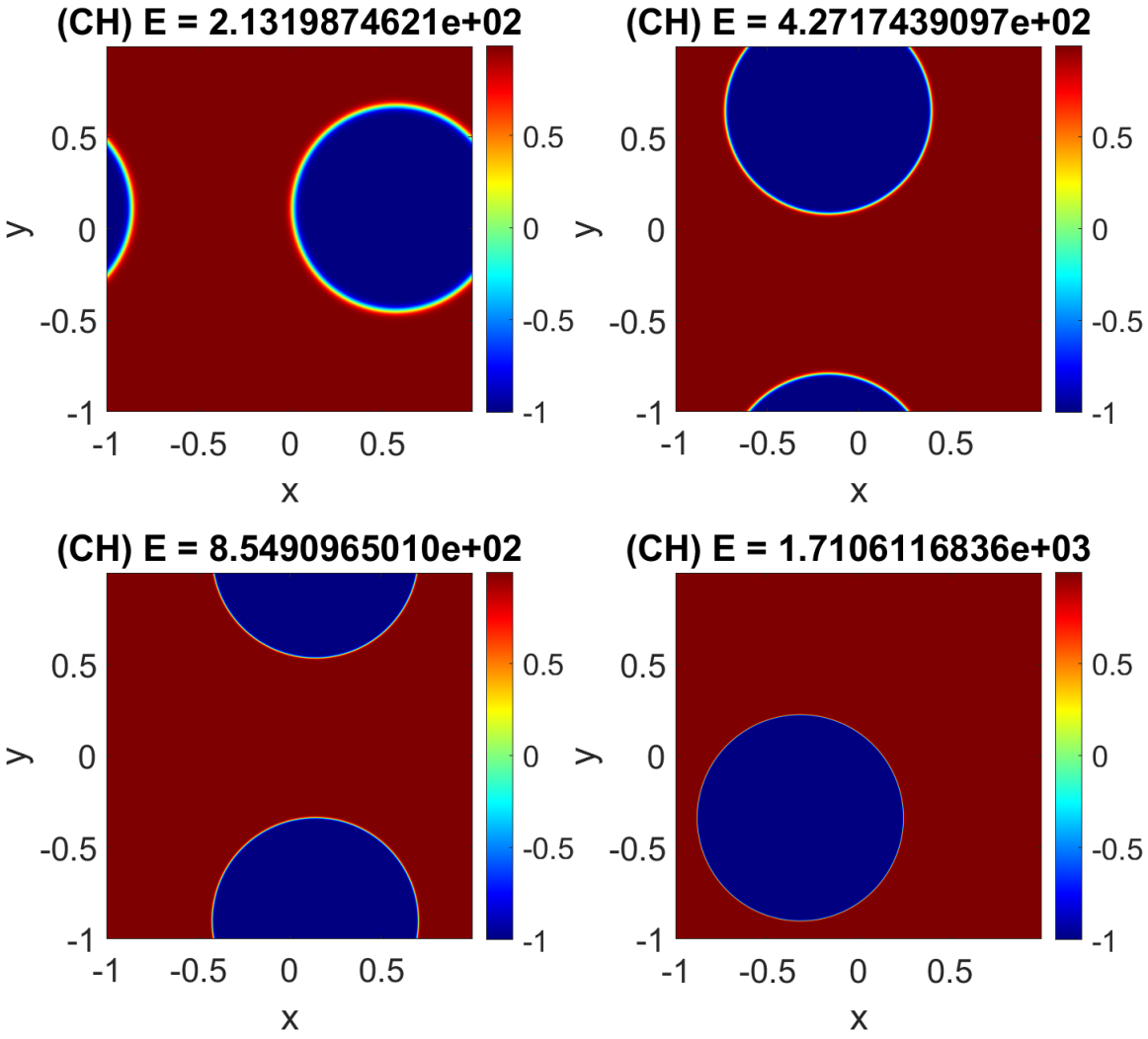


Figure 3.2: Corresponding contour plots of the steady-state solutions u_g for the 2D CH equation obtained by the PCG method (Top left : $\epsilon = \frac{1}{64}$; Top right : $\epsilon = \frac{1}{128}$; Bottom left: $\epsilon = \frac{1}{256}$; Bottom right: $\epsilon = \frac{1}{512}$).

3.4.2 Numerical Experiments in 3D

In this section, we apply the PCG method to compute some 3D challenging problems. The computational domain is $\mathcal{D} = [-1, 1]^3$. For the AC equation, the initial approximations are

$$u_0(x, y, z) = \sin(2\pi x) + 0.001\cos(16\pi x), \tag{3.33}$$

in order to avoid the trivial steady-state solutions. For the CH equation, the initial approximations are random entries uniformly distributed on $[0, 1]$, generated by MATLAB's `rand` function with the fixed seed 1. Other details are the same as in the 2D domains. We summarize the results for the AC and the CH equations in Table 3.4 and 3.5, respectively. Also, Figure 3.3 and 3.4 show the corresponding isosurface plot $|u_g|^2 = 1$. We can see that the PCG method still performs very efficiently extending to the 3D domains.

Table 3.4: Performance of the PCG method for the 3D AC equation.

ϵ	N	PCG		Energy
		Iter	Time	
1/32	2097152	29	6.50	482.71822924
1/64	16777216	85	163.63	965.43646020
1/128	134217728	548	7801.12	1930.8729152

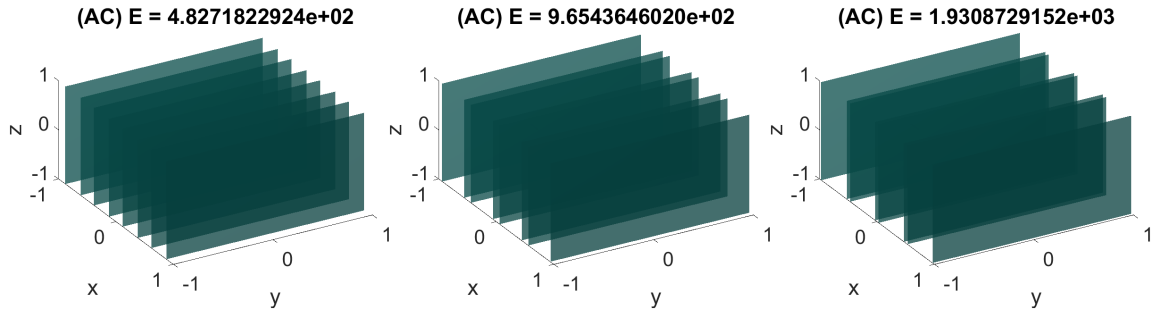


Figure 3.3: Corresponding isosurface plots of the steady-state solutions $|u_g|^2 = 1$ for the 3D AC equation obtained by the PCG method in Table 3.4 (Left : $\epsilon = \frac{1}{32}$; Middle : $\epsilon = \frac{1}{64}$; Right: $\epsilon = \frac{1}{128}$).

Table 3.5: Performance of the PCG method for the 3D CH equation.

ϵ	N	PCG		Energy
		Iter	Time	
1/32	2097152	321	106.84	212.12420022
1/64	16777216	962	2454.06	459.94897753
1/128	134217728	18223	368850.70	854.07857874

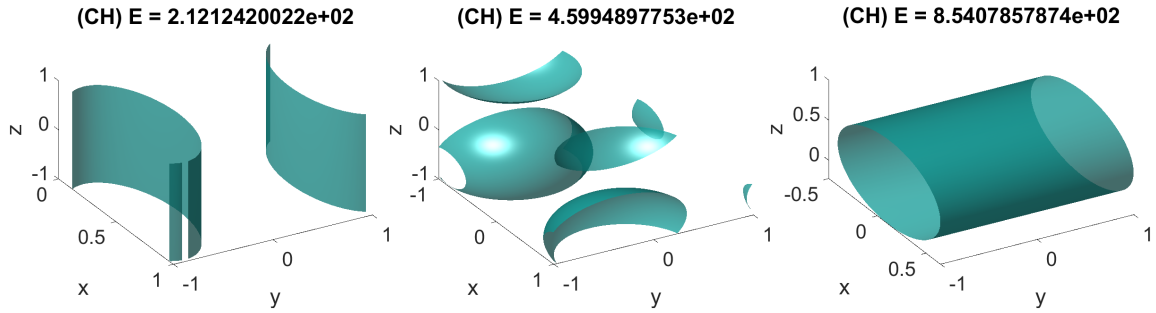


Figure 3.4: Corresponding isosurface plots of the steady-state solutions $|u_g|^2 = 1$ for the 3D CH equation obtained by the PCG method in Table 3.5 (Left : $\epsilon = \frac{1}{32}$; Middle : $\epsilon = \frac{1}{64}$; Right: $\epsilon = \frac{1}{128}$).

Chapter 4

Computing ground states of Bose-Einstein condensation

In this chapter, we propose and study an improved PCG method for computing the ground state of rotational BEC (1.15). Our new method makes exclusive use of real arithmetic to fully exploit the special structure of the problem and realizes fast numerical linear algebra computations. We discuss a simple approach to achieve fast energy evaluations for many different step sizes along the search direction at little additional cost, which enables exact line search efficiently. In addition, we derive the explicit expression of the discrete Hessian operator of the energy $E(\phi)$ in real arithmetic and propose an approximate shifted Hessian preconditioner that is quite efficient for tackling high nonlinearity strength η and high rotational speed Ω. We show that the preconditioned Hessian with our ideal preconditioner has favorable eigenvalue distributions independent of the mesh size h. Therefore, given a rotational BEC problem in a specified domain, the PCG method with our ideal preconditioner is expected to exhibit mesh size-independent asymptotic convergence behavior.

The remainder of this chapter is organized as follows. In Section 4.1, we present a detailed description of our method. In Section 4.2, we derive the discrete Hessian operator of energy functional $E(\phi)$ and provide the preconditioning strategy in practice. Section 4.3 provides an accurate and efficient method to enable fast energy evaluation and exact line search. We study the eigenvalue distribution of the preconditioned Hessian with our ideal preconditioner in Section 4.4. Section 4.5 provides numerical results in 2D and 3D domains to validate our new developments.

4.1 The preconditioned conjugate gradient (PCG) method in real arithmetic

To develop an efficient solver for problems involving complex numbers, an important strategy in numerical linear algebra is to fully use real arithmetic whenever appropriate. Since $E_{\eta,\Omega}$ in (1.17) is real even though ϕ is complex, computation in real arithmetic is desired, especially for optimization algorithms where $E_{\eta,\Omega}$ needs to be evaluated many times. To the best of our knowledge, nearly all existing algorithms for computing BEC ground states use complex arithmetic, with an exception in [58] that requires solutions of a long sequence of large linear systems that arise in a special nonlinear inverse iteration to solve the nonlinear eigenvalue problem (1.18).

First, we will reformulate the BEC problem in real arithmetic. To develop new methods in real arithmetic, let $\phi = \phi_r + i\phi_g \in \mathbb{C}^n$, where ϕ_r and ϕ_g are the real and imaginary parts of ϕ, with $\|\phi\|_{\ell^2}^2 = \|\phi_r\|_{\ell^2}^2 + \|\phi_g\|_{\ell^2}^2 = 1$. Define $L_s = -\frac{1}{2}L_p + \mathrm{diag}(V)$(symmetric positive definite). The energy (1.17) in real arithmetic has the form

$$E_{\eta,\Omega} = \left[\phi_r^T L_s \phi_r + \phi_g^T L_s \phi_g + \frac{\eta}{2}(\phi_r^2 + \phi_g^2)^T(\phi_r^2 + \phi_g^2) + 2\Omega\phi_r^T L_\omega \phi_g\right] h^d, \tag{4.1}$$

with $\|\phi\|_{\ell^2} = h^{d/2}\|\phi\|_2 = 1$. Note that ϕ_r^2 is the column vector whose entries are the squares of those of ϕ_r, and ϕ_g^2 is defined similarly. The evaluation of (4.1) takes only half of the arithmetic cost needed to evaluate (1.17) in complex arithmetic. Note that the evaluation of (1.17) in complex arithmetic [11] did not take advantage of the special structure of $-L_p$ and L_ω, which might involve more round-off errors, give a complex energy value with a small imaginary part, and could make the final converged energy $E_{\eta,\Omega}(\phi)$ less accurate in high accuracy demand.

In order to employ the PCG method, it is necessary to obtain the gradient of $E_{\eta,\Omega}$ (4.1). Note that the energy expression of $E_{\eta,\Omega}$ (1.17) or (4.1) is valid under the normalization constraint $\|\phi\|_{\ell^2} = 1$. But we may disregard it and derive its gradient formally. The gradient of $E_{\eta,\Omega}$ (4.1) with respect to $\phi = \begin{pmatrix}\phi_r^T & \phi_g^T\end{pmatrix}^T$ is

$$\frac{\partial E_{\eta,\Omega}}{\partial \phi} = 2\begin{pmatrix} L_s\phi_r + \eta\,\mathrm{diag}(\phi_r^2 + \phi_g^2)\phi_r + \Omega L_\omega \phi_g \\ L_s\phi_g + \eta\,\mathrm{diag}(\phi_r^2 + \phi_g^2)\phi_g - \Omega L_\omega \phi_r \end{pmatrix} h^d. \tag{4.2}$$

We can disregard the factor h^d and keep the direction $\frac{\partial E_{\eta,\Omega}}{\partial \phi}$. Since ϕ is restricted on the sphere $\|\phi\|_{\ell^2} = 1$, the *effective* gradient is the component of (4.2) that is orthogonal to ϕ :

$$r_{\eta,\Omega} = \begin{pmatrix} L_s\phi_r + \eta \mathrm{diag}(\phi_r^2 + \phi_g^2)\phi_r + \Omega L_\omega \phi_g \\ L_s\phi_g + \eta \mathrm{diag}(\phi_r^2 + \phi_g^2)\phi_g - \Omega L_\omega \phi_r \end{pmatrix} - \lambda_{\eta,\Omega} \begin{pmatrix} \phi_r \\ \phi_g \end{pmatrix}, \tag{4.3}$$

where

$$\lambda_{\eta,\Omega} = \left[\phi_r^T L_s \phi_r + \phi_g^T L_s \phi_g + \eta(\phi_r^2 + \phi_g^2)^T(\phi_r^2 + \phi_g^2) + 2\Omega \phi_r^T L_\omega \phi_g\right] h^d \tag{4.4}$$

such that $\phi^T r_{\eta,\Omega} = 0$. Note that (4.3), up to a scaling factor, can also be derived by differentiating the scaling invariant energy $E(\phi)$ as performed in Section 4.2. We call $\lambda_{\eta,\Omega}$ the nonlinear Rayleigh functional of ϕ, which approximates the desired eigenvalue in (1.18). Also, $\lambda_{\eta,\Omega}$ represents the *chemical potential* [16] and

$$\lambda_{\eta,\Omega} = E_{\eta,\Omega} + \frac{\eta}{2}(\phi_r^2 + \phi_g^2)^T(\phi_r^2 + \phi_g^2)h^d = E_{\eta,\Omega} + E_{int}, \tag{4.5}$$

where $E_{int} = \frac{\eta}{2}(\phi_r^2 + \phi_g^2)^T(\phi_r^2 + \phi_g^2)h^d$ is the *interaction energy*. Therefore, $r_{\eta,\Omega}$ is the eigenresidual associated with ϕ of the nonlinear eigenvalue problem (1.18).

Now, a nonlinear preconditioned conjugate gradient (PCG) method in real arithmetic could be employed in an effort to find the global minimizer of the energy functional (4.1). Suppose we work with a generic preconditioner M. The standard search direction in nonlinear PCG is

$$d_{(k)} = -M^{-1} r_{(k)} + \beta_{(k)} d_{(k-1)}, \tag{4.6}$$

with the Fletcher–Reeves update [52]

$$\beta_{(k)} = \frac{\langle r_{(k)}, M^{-1} r_{(k)} \rangle}{\langle r_{(k-1)}, M^{-1} r_{(k-1)} \rangle}.$$

At iteration k, a regular update formula for $\phi_{(k+1)}$ in PCG could be

$$\phi_{(k+1)} = \phi_{(k)} \cos(\theta_{(k)}) + p_{(k)} \sin(\theta_{(k)}), \tag{4.7}$$

Here, $p_{(k)}$ is the modified search direction, which is orthogonal to $\phi_{(k)}$ in complex arithmetic and normalized in the standard ℓ^2-space.

Given two complex vectors $\widehat{d_{(k)}} = d_{(k)r} + id_{(k)g}$ and $\widehat{\phi_{(k)}} = \phi_{(k)r} + i\phi_{(k)g} \in \mathbb{C}^n$, the real representation of $\widehat{d_{(k)}}$ and $\widehat{\phi_{(k)}}$ are $d_{(k)} = \begin{pmatrix} d_{(k)r}^T & d_{(k)g}^T \end{pmatrix}^T$ and $\phi_{(k)} = \begin{pmatrix} \phi_{(k)r}^T & \phi_{(k)g}^T \end{pmatrix}^T$, respectively. Then, the orthogonalization of $\widehat{d_{(k)}}$ against $\widehat{\phi_{(k)}}$ in complex arithmetic which gives the result complex vector $\widehat{p_{(k)}} = p_{(k)r} + ip_{(k)g}$, can be done by their real representations as follows:

$$p_{(k)} = d_{(k)} - W(W^TW)^{-1}W^Td_{(k)}, \tag{4.8}$$

where $p_{(k)} = \begin{pmatrix} p_{(k)r} \\ p_{(k)g} \end{pmatrix}$ and $W = \begin{pmatrix} \phi_{(k)r} & -\phi_{(k)g} \\ \phi_{(k)g} & \phi_{(k)r} \end{pmatrix}$. Moreover, the normalization condition $\|p_{(k)}\|_{\ell^2} = 1$ can be easily done by

$$p_{(k)} = p_{(k)}/(h^{\frac{d}{2}}\|p_{(k)}\|_2). \tag{4.9}$$

Note that (4.8) and (4.9) ensure that $\phi_{(k+1)}$ obtained from (4.7) satisfies the normalization constraint such that $\|\phi_{(k+1)}\|_{\ell^2} = 1$ for any $\theta_{(k)}$.

An outline of the nonlinear PCG is given in Algorithm 3.

Algorithm 3 The preconditioned conjugate gradient method.

1: Start with an initial approximation $\phi_{(0)}$ with $\|\phi_{(0)}\|_{\ell^2} = 1$.
2: **while** not converged **do**
3: $\lambda_{(k)} = \lambda_{\eta,\Omega}(\phi_{(k)})$ (see (4.4)).
4: $r_{(k)} = r_{\eta,\Omega}(\phi_{(k)})$ (see (4.3)).
5: $\beta_{(k)} = \frac{\langle r_{(k)}, M^{-1}r_{(k)}\rangle}{\langle r_{(k-1)}, M^{-1}r_{(k-1)}\rangle}$
6: $d_{(k)} = -M^{-1}r_{(k)} + \beta_{(k)}d_{(k-1)}$
7: $p_{(k)} = d_{(k)} - W(W^TW)^{-1}W^Td_{(k)}$
8: $p_{(k)} = p_{(k)}/(h^{\frac{d}{2}}\|p_{(k)}\|_2)$
9: $\theta_{(k)} = \arg\min_\theta E(\phi_{(k)}\cos(\theta) + p_{(k)}\sin(\theta))$
10: $\phi_{(k+1)} = \phi_{(k)}\cos(\theta_{(k)}) + p_{(k)}\sin(\theta_{(k)})$
11: $k = k + 1$
12: **end while**

4.2 Problem-dependent Hessian preconditioner

One critical problem in the nonlinear PCG is to design a good preconditioner, which can significantly reduce the runtime. In general, the preconditioner for PCG near convergence should be an approximation to the Hessian of the objective function. The most well-known case is to apply the PCG method to find the solution of a linear system $Ax = b$, where A is symmetric and positive definite. In this case, the objective function is $f(x) = \frac{1}{2}x^T Ax - b^T x$, which leads to the gradient of $f(x)$ is $\nabla f(x) = Ax - b$ and the Hessian of $f(x)$ is A. Then, minimizing $f(x)$ is equivalent to solving the linear system $Ax = b$. The preconditioner for PCG to solve this linear system should be an approximation to A. Also, it is reasonable to expect to use an approximation to the Hessian of the energy (4.1) as the preconditioner for the nonlinear PCG method.

4.2.1 Derivation of the discrete Hessian operator

In this section, we will derive the explicit expression of the discrete Hessian operator for the energy functional $E_{\eta,\Omega}$ (4.1) based on the real arithmetic and introduce the preconditioning strategy in practice. It is crucial to point out the discrete Hessian operator of a real-valued scalar function of n complex variables must be a linear operator that operates on a vector of $2n$ degrees of freedom [69]. In particular, this means that such a Hessian operator may not be represented correctly as a complex matrix of order n. In [69], three definitions of the Hessian are given for such a function based on the real arithmetic or complex arithmetic, and all these Hessian matrices are of order $2n$. For the BEC problems, with our use of real arithmetic, it is also natural to derive the discrete Hessian operator of the energy functional (1.17) as a real symmetric matrix of order $2n$. In order to derive the full expression of discrete Hessian operator of $E_{\eta,\Omega}$, we will follow the scheme in [58] to absorb the normalization constraint to rewrite (4.1) in a form that is invariant with respect to the scaling of the wave function ϕ.

Assume that $\phi = \begin{pmatrix} \phi_r^T & \phi_g^T \end{pmatrix}^T$, which leads to the equivalent expression of (4.1)

$$E(\phi) = \frac{\phi^T A\phi}{\phi^T\phi} + \frac{\eta}{2}\frac{\phi^T B(\phi)\phi}{h^d(\phi^T\phi)^2}, \tag{4.10}$$

where

$$A = \begin{pmatrix} L_s & \Omega L_\omega \\ -\Omega L_\omega & L_s \end{pmatrix} \quad \text{and} \quad B(\phi) = \begin{pmatrix} \text{diag}(\phi_r^2 + \phi_g^2) & 0 \\ 0 & \text{diag}(\phi_r^2 + \phi_g^2) \end{pmatrix}.$$

Here, A is real symmetric and $E(\phi)$ satisfies the scaling invariant property, i.e., $E(\alpha\phi) = E(\phi)$ for any $\alpha \in \mathbb{R}\backslash\{0\}$. Theorem 4.2.1 provides the complete expression of the gradient and the Hessian for $E(\phi)$ in (4.10).

Theorem 4.2.1. *The gradient and Hessian for $E(\phi)$ (4.10) are given by*

$$\frac{\partial E(\phi)}{\partial \phi} = \frac{2}{\phi^T\phi}\left(A(\phi)\phi - \lambda(\phi)\phi\right) \tag{4.11}$$

and

$$\begin{aligned} \frac{\partial^2 E(\phi)}{\partial \phi^2} = \frac{2}{\phi^T\phi}\Bigg\{ & A + \frac{\eta}{h^d \phi^T\phi}\begin{pmatrix} \text{diag}(3\phi_r^2 + \phi_g^2) & 2\text{diag}(\phi_r\phi_g) \\ 2\text{diag}(\phi_r\phi_g) & \text{diag}(\phi_r^2 + 3\phi_g^2) \end{pmatrix} - \lambda(\phi) I \\ & -2A\frac{\phi\phi^T}{\phi^T\phi} - 2\frac{\phi\phi^T}{\phi^T\phi}A - 4\eta\frac{B(\phi)}{h^d\phi^T\phi}\frac{\phi\phi^T}{\phi^T\phi} - 4\eta\frac{\phi\phi^T}{\phi^T\phi}\frac{B(\phi)}{h^d\phi^T\phi} \\ & +4\frac{\phi^T A\phi}{\phi^T\phi}\frac{\phi\phi^T}{\phi^T\phi} + 6\eta\frac{\phi\phi^T}{\phi^T\phi}\frac{\phi^T B(\phi)\phi}{h^d(\phi^T\phi)^2}\Bigg\}, \end{aligned} \tag{4.12}$$

where

$$A(\phi) = A + \eta\frac{B(\phi)}{h^d\phi^T\phi} \;\; \textit{and} \;\; \lambda(\phi) = \frac{\phi^T A\phi}{\phi^T\phi} + \eta\frac{\phi^T B(\phi)\phi}{h^d(\phi^T\phi)^2}.$$

Also, if $\phi = \begin{pmatrix} \phi_r \\ \phi_g \end{pmatrix}$ is a stationary point of $E(\phi)$ such that

$$\frac{\partial E(\phi)}{\partial \phi} = \frac{2}{\phi^T\phi}\left(A(\phi)\phi - \lambda(\phi)\phi\right) = 0,$$

we have

$$\frac{\partial^2 E(\phi)}{\partial \phi^2}\phi = 0, \;\; \textit{and} \;\; \frac{\partial^2 E(\phi)}{\partial \phi^2}\widehat{\phi} = 0,$$

where $\widehat{\phi} = \begin{pmatrix} -\phi_g \\ \phi_r \end{pmatrix}$*; that is,* ϕ *and* $\widehat{\phi}$ *are the eigenvectors of* $\frac{\partial^2 E(\phi)}{\partial \phi^2}$ *associated with the zero eigenvalue.*

Proof. The proof is given in the Appendix. □

Note that the gradient from Theorem 4.2.1 is consistent with (4.3) up to a scaling factor. Also, we can see that the discrete Hessian operator of the energy $E_{\eta,\Omega}(\phi)$ should be of order $2n$. Now, we are ready to introduce a shifted Hessian preconditioner for the nonlinear PCG method.

4.2.2 Preconditioning strategy

For the nonlinear PCG, it is crucial to apply the preconditioner efficiently. For the ground state solution, i.e, the global minimizer of $E_{\eta,\Omega}$, the first order optimality condition is $r_{\eta,\Omega} = 0$, and the second order optimality condition is $H_{\eta,\Omega} \succeq 0$ (positive semidefinite) with the null space spanned by the ground state solution $\phi = \begin{pmatrix} \phi_r^T & \phi_g^T \end{pmatrix}^T$ and $\widehat{\phi} = \begin{pmatrix} -\phi_g^T & \phi_r^T \end{pmatrix}^T$. Suppose that P is the orthogonal projector with null space spanned by the ground state $\phi = \begin{pmatrix} \phi_r^T & \phi_g^T \end{pmatrix}^T$ and $\widehat{\phi} = \begin{pmatrix} -\phi_g^T & \phi_r^T \end{pmatrix}^T$, i.e., $P = I - W(W^TW)^{-1}W^T$, where $W = \begin{pmatrix} \phi_r & -\phi_g \\ \phi_g & \phi_r \end{pmatrix}$.

With the normalization constraint $\phi^T \phi h^d = 1$, we obtain $W^TW = \frac{1}{h^d} I$, so that $P = I - h^d WW^T$. Since that $P\phi = \phi^T P = 0$, the low-rank updates in (4.12) can be cancelled out by multiplying $\frac{\partial^2 E(\phi)}{\partial \phi^2}$ on both sides by P. That is,

$$P \frac{\partial^2 E(\phi)}{\partial \phi^2} P = \frac{2}{\phi^T \phi} P \left\{ A + \frac{\eta}{h^d \phi^T \phi} \begin{pmatrix} \mathrm{diag}(3\phi_r^2 + \phi_g^2) & 2\mathrm{diag}(\phi_r \phi_g) \\ 2\mathrm{diag}(\phi_r \phi_g) & \mathrm{diag}(\phi_r^2 + 3\phi_g^2) \end{pmatrix} - \lambda(\phi) I \right\} P.$$

Therefore, we can define the *effective* Hessian of $E_{\eta,\Omega}$ (4.1), i.e.,

$$H_{\eta,\Omega} := P H_p P = P \left\{ \begin{pmatrix} L_s + \eta \mathrm{diag}(3\phi_r^2 + \phi_g^2) & \Omega L_\omega + 2\eta \mathrm{diag}(\phi_r \phi_g) \\ -\Omega L_\omega + 2\eta \mathrm{diag}(\phi_r \phi_g) & L_s + \eta \mathrm{diag}(\phi_r^2 + 3\phi_g^2) \end{pmatrix} - \lambda I_{2n} \right\} P, \tag{4.13}$$

where $\phi_r \phi_g$ is the column vector whose entries are the product of those of ϕ_r and ϕ_g. Moreover, the projector P adopted can avoid the potential stagnation of the 'correction direction' that occurred in Davidson-type eigensolvers [86].

To speed up the convergence of our optimization methods, we define the *shifted Hessian preconditioner* based on (4.13) as

$$M_{\eta,\Omega} := PM_pP = P\left\{\begin{pmatrix} L_s + \eta\mathrm{diag}(3\phi_r^2 + \phi_g^2) & \Omega L_\omega + 2\eta\mathrm{diag}(\phi_r\phi_g) \\ -\Omega L_\omega + 2\eta\mathrm{diag}(\phi_r\phi_g) & L_s + \eta\mathrm{diag}(\phi_r^2 + 3\phi_g^2) \end{pmatrix} - (\lambda - \sigma)I_{2n}\right\}P. \quad (4.14)$$

The shift $\sigma > 0$ is chosen such that $M_p \succ 0$ (positive definite) near the ground state and ensures that incomplete Cholesky factorization of M_p can be done successfully. A smaller σ lead to M_p closer to the effective Hessian $H_{\eta,\Omega}$ (more effective preconditioning), whereas a larger σ makes M_p less close to $H_{\eta,\Omega}$ (less effective). In practice, σ should be chosen to strike a balance between the chance of success of incomplete Cholesky factorization and the effectiveness of preconditioning. In our numerical experiments, we let $\sigma = (E_{\eta,\Omega} + \lambda_{\eta,\Omega})/2$ for the current iterate $\phi_{(k)}$ by default, though this choice can be easily changed if necessary.

Given the real formulation of our proposed preconditioner $M_{\eta,\Omega}$ (4.14), one might wonder if we could find a complex Hermitian $\widehat{M}_{\eta,\Omega}$ of order n, and form the vector u in complex arithmetic $\widehat{u} = u_r + iu_g$, such that $M_{\eta,\Omega}^{-1}u = M_{\eta,\Omega}^{-1}\begin{pmatrix} u_r^T & u_g^T \end{pmatrix}^T$ and $\widehat{M}_{\eta,\Omega}^{-1}\widehat{u}$ represent the same vector in real and complex arithmetic, respectively. This is equivalent to find a complex Hermitian $\widehat{M}_p$ of order n such that $M_p^{-1}u$ and $\widehat{M}_p^{-1}\widehat{u}$ represent the same vector in real and complex arithmetic, respectively. It can be shown this is impossible. Suppose $M_p^{-1}u = v$ and $\widehat{M}_p^{-1}\widehat{u} = \widehat{v}$ such that $\widehat{v} = v_r + iv_g$, then M_pv and $\widehat{M}_p\widehat{v}$ represent the same vector in real and complex arithmetic, respectively. Let $\widehat{M}_p = \mathrm{Re}(\widehat{M}_p) + i\,\mathrm{Im}(\widehat{M}_p)$, then we have $M_pv = \begin{pmatrix} \mathrm{Re}(\widehat{M}_p) & -\mathrm{Im}(\widehat{M}_p) \\ \mathrm{Im}(\widehat{M}_p) & \mathrm{Re}(\widehat{M}_p) \end{pmatrix} v$, which leads to a contradiction, since the $(1,2)$ and $(2,1)$ blocks of M_p are not opposite of each other unless $\eta = 0$. Real arithmetic computation is essential to enable a wide range of options to approximate the action of the Hessian H_p, for both Newton-like and preconditioner conjugate gradient-like methods for the minimization of $E_{\eta,\Omega}$.

A good preconditioner should be applied efficiently. The difficulty of applying the Hessian preconditioner depends on the discretization scheme used. Under the Fourier pseudo-spectral discretization scheme we adopt, the discrete Hessian operator (4.14) is fully dense, and geometric multigrid (GMG) is a reasonable method to approximate the action of $M_{\eta,\Omega}^{-1}$ on vectors. A more efficient alternative, however, is to construct the shifted Hessian operator (4.14) in finite difference

discretization based on the same uniform mesh; this leads to a sparse approximation to the true discrete Hessian operator (4.12) in Fourier pseudo-spectral scheme, to which incomplete Cholesky factorization can be applied efficiently. This preconditioning strategy is reasonable in this setting, since the action of preconditioning usually does not need to be computed to high accuracy. In practice, we apply the 8th order finite difference approximations to form $M_{\eta,\Omega}$ (4.14), which seems accurate enough to approximate the true discrete Hessian operator (4.12) given that the wave function ϕ has complex pattern of vortexes. In this way, the *shifted Hessian preconditioner* actually used is sparse and can be applied efficiently. For large problems, as exact matrix factorizations are prohibitive, we apply incomplete Cholesky factorization with fill-reducing permutations such as approximate minimal degree ordering [4] and an appropriate drop tolerance. Nevertheless, we still cannot afford to perform a new factorization at each step of the nonlinear PCG. To lower the computational cost, we keep the same preconditioner for a certain number of PCG steps before performing a new factorization. Compared with the state-of-the-art Combined preconditioner proposed in [11], our Hessian preconditioner is not cheap to apply. To use our preconditioner when it is expected to be effective, we propose a two-stage preconditioning strategy to further lower the computational cost. We use the Combined preconditioner at the first stage and switch to our Hessian preconditioner at the second stage. The timing of the switch depends on the relative difference between the energy of two consecutive PCG iterations. In our numerical experiments, we found that this strategy performs pretty well. Generally, the Combined preconditioner is effective to help PCG proceed closer to the final minimized energy, but it tends to struggle or even stagnate near the convergence, whereas our Hessian preconditioner can help PCG converge to the final energy more rapidly in a robust manner.

4.3 Fast energy functional evaluation and exact line search

For gradient-based optimization methods, it is a common practice to perform an approximate line search following the Armijo-Goldstein or Wolfe conditions [72], since exact line search is prohibitive for large problems. There are a few well-known counterexamples, such as PCG for solving a symmetric positive definite (SPD) linear system and computing the lowest eigenvalue(s) of an SPD matrix, as exact line search can be done efficiently with explicit formula for the optimal step size [55] or by the Rayleigh-Ritz projection [64]. The state-of-the-art variant of PCG for rotational BEC [11] performs line search by approximating $E_{\eta,\Omega}$ by a quadratic function and some complex

methodologies based on different conditions and certain default values not explicitly specified. Fortunately, we find that fast exact line search can be enabled, *without* repeated evaluations of the energy functional at different step sizes in the original problem dimension n.

Specifically, let $\phi_{(k)}$ be the current ground state approximation with $\|\phi_{(k)}\|_{\ell^2} = 1$, and $d_{(k)}$ be a search direction. We orthogonalize $d_{(k)}$ against $\phi_{(k)}$ then normalize it in $\|\cdot\|_{\ell^2}$ norm into $p_{(k)}$ following the process introduced in Section 4.1. Then, the new iterate is $\phi_{(k+1)} = \phi_{(k)}\cos(\theta_{(k)}) + p_{(k)}\sin(\theta_{(k)})$, where $\theta_{(k)}$ is the minimizer $E(\phi_{(k)}\cos(\theta) + p_{(k)}\sin(\theta))$. By construction, we know $\|\phi_{(k+1)}\|_{\ell^2} = 1$. Consider the objective function $E_{\eta,\Omega}$ (4.1), substitute $\phi = \begin{pmatrix}\phi_r^T & \phi_g^T\end{pmatrix}^T$ with $\begin{pmatrix}\phi_{(k)r}^T & \phi_{(k)g}^T\end{pmatrix}^T \cos\theta + \begin{pmatrix}p_{(k)r}^T & p_{(k)g}^T\end{pmatrix}^T \sin\theta$ into $E_{\eta,\Omega}$. Then, by direct algebraic evaluation, we get

$$\begin{aligned} E_{\eta,\Omega}(\phi_{(k)}\cos\theta + p_{(k)}\sin\theta) = \big[&w(\theta)^T L_{s(k)} w(\theta) + 2\Omega w(\theta)^T L_{\omega(k)} w(\theta) + \\ &\frac{\eta}{2}\left(c_1\cos^4\theta + c_2\cos^3\theta\sin\theta + c_3\cos^2\theta\sin^2\theta + c_4\cos\theta\sin^3\theta + c_5\sin^4\theta\right)\big] h^d, \end{aligned} \tag{4.15}$$

where

$$\begin{aligned} w(\theta) &= \begin{pmatrix}\cos\theta & \sin\theta\end{pmatrix}^T,\ L_{\omega(k)} = \begin{pmatrix}\phi_{(k)r} & p_{(k)r}\end{pmatrix}^T L_\omega \begin{pmatrix}\phi_{(k)g} & p_{(k)g}\end{pmatrix} \in \mathbb{R}^{2\times 2}, \\ L_{s(k)} &= \begin{pmatrix}\phi_{(k)r} & p_{(k)r}\end{pmatrix}^T L_s \begin{pmatrix}\phi_{(k)r} & p_{(k)r}\end{pmatrix} + \begin{pmatrix}\phi_{(k)g} & p_{(k)g}\end{pmatrix}^T L_s \begin{pmatrix}\phi_{(k)g} & p_{(k)g}\end{pmatrix} \in \mathbb{R}^{2\times 2}, \end{aligned}$$

and

$$\begin{aligned} &c_1 = (\phi_{(k)r}^2 + \phi_{(k)g}^2)^T(\phi_{(k)r}^2 + \phi_{(k)g}^2),\ c_2 = 4(\phi_{(k)r}^2 + \phi_{(k)g}^2)^T(\phi_{(k)r}p_{(k)r} + \phi_{(k)g}p_{(k)g}), \\ &c_3 = 4(\phi_{(k)r}p_{(k)r} + \phi_{(k)g}p_{(k)g})^T(\phi_{(k)r}p_{(k)r} + \phi_{(k)g}p_{(k)g}) + 2(\phi_{(k)r}^2 + \phi_{(k)g}^2)^T(p_{(k)r}^2 + p_{(k)g}^2), \\ &c_4 = 4(\phi_{(k)r}p_{(k)r} + \phi_{(k)g}p_{(k)g})^T(p_{(k)r}^2 + p_{(k)g}^2),\ c_5 = (p_{(k)r}^2 + p_{(k)g}^2)^T(p_{(k)r}^2 + p_{(k)g}^2). \end{aligned}$$

Here, $\phi_{(k)r}$ and $\phi_{(k)g}$ stand for the real part and imaginary part of $\phi_{(k)}$, respectively; $p_{(k)r}$ and $p_{(k)g}$ are defined similarly, and $\phi_{(k)r}p_{(k)r}$ stands for the column vector whose entries are the product of those of $\phi_{(k)r}$ and $p_{(k)r}$; $\phi_{(k)r}^2$, $\phi_{(k)g}^2$, $p_{(k)r}^2$ and $p_{(k)g}^2$ can be defined similarly.

The key observation is that it only takes 6 matrix vector multiplications of order n, 18 vector inner product of order n, 6 element-wise vector multiplications of order n and 3 vector additions of order n to obtain $L_{s(k)}$, $L_{\omega(k)} \in \mathbb{R}^{2\times 2}$, and the scalars c_i $(1 \le i \le 5)$, *no more* computation in the original problem dimension n is needed. Now $E_{\eta,\Omega}(\phi_{(k)}\cos\theta + p_{(k)}\sin\theta) : \mathbb{R} \to \mathbb{R}$ can be evaluated

for any and as many values of θ as needed at little arithmetic cost. We can afford to perform a numerical exact line search to minimize $E_{\eta,\Omega}(\phi_{(k)}\cos\theta + p_{(k)}\sin\theta)$, or find the minimizer by forming $\frac{d}{d\theta}E_{\eta,\Omega}(\phi_{(k)}\cos\theta + p_{(k)}\sin\theta) = 0$ in closed form and solving it for θ. In our implementation, we use MATLAB's `fminsearch` function to find the optimal θ, which can be done rapidly without additional work on dimension n. Note that this procedure is equivalent to the Rayleigh-Ritz procedure in many iterative methods to solve linear or linearized symmetric eigenproblems for the lowest eigenvalues.

Similarly, the fast exact search can be applied to the locally optimal preconditioned conjugate gradient method (LOPCG) [63], which is very successful in nonlinear eigenproblems. Assume that we have determined 2 search directions $p_{(k)}, f_{(k)}$ such that $\|p_{(k)}\|_{\ell^2}^2 = \|f_{(k)}\|_{\ell^2}^2 = 1$, and $\phi_{(k)}, p_{(k)}$, $f_{(k)}$ are pairwise orthogonal. To determine the new iterate $\phi_{(k+1)}$ for which the energy functional is minimized, define $\phi_{(k+1)} = \phi_{(k)}\cos\theta + p_{(k)}\sin\theta\cos\gamma + f_{(k)}\sin\theta\sin\gamma$. The simplified expression $E_{\eta,\Omega}(\phi_{(k+1)})$ can be derived and one only needs to compute $L_{s(k)}, L_{\Omega(k)} \in \mathbb{R}^{3\times 3}$ and 15 scalar coefficients once to evaluate $E_{\eta,\Omega}$ all values of (θ,γ) efficiently. However, we found that adding more directions in the search subspace does not yield significant gain in runtime consistently.

In [11, 12], a quadratic approximation line search is provided. Without the details of the specified parameters, we find that it is not easy to achieve the fast convergence of the nonlinear PCG. Here, we provide a modified quadratic approximation line search. Assuming $\epsilon_{(k)}$ is the eigenresidual at the step k, we can approximate $E_{\eta,\Omega}$ by a quadratic function, which is evaluated at $\theta_{(k)} = 0$, $\epsilon_{(k)}/2$ and $\epsilon_{(k)}$, respectively. Then, we can use the minimizer $\theta_{(k)}^{opt}$ of the corresponding quadratic function as a trial step size. If the energy $E_{\eta,\Omega}(\theta_{(k)}^{opt})$ is decreased, we accept this step. Otherwise, we reject this step, decrease the interpolation step sizes by a factor of 2 (e.g., 0, $\epsilon_{(k)}/4$ and $\epsilon_{(k)}/2$), and try again, until the energy is decreased, which ensures that $\theta_{(k)}$ is small enough. However, the performance of the nonlinear PCG can still be affected by the choice of the three interpolation points. Furthermore, a backtracking line search with Armijo–Goldstein condition [13] can also be employed here. Compared with these line search methods, our exact line search can avoid tuning parameters and help the nonlinear PCG converge more robustly.

4.4 Expected behavior of PCG near convergence

Given the well-known results about the convergence properties of the conjugate gradient method for preconditioned linear systems [79] and the steepest descent methods for nonlinear con-

strained minimization [1], the convergence of the nonlinear PCG relies on the properties of the preconditioned Hessian operator. In this section, we show the pattern of the spectrum for the preconditioned Hessian with our shifted Hessian preconditioner (4.14), such that the nonlinear PCG is expected to converge quickly when approaching convergence.

The preconditioned Hessian operator with the *shifted Hessian preconditioner* is defined as, in the way suggested in [11],

$$PM_{\eta,\Omega}^{-1}\frac{\partial^2 E(\phi)}{\partial \phi^2}P. \tag{4.16}$$

Suppose that $M_{\eta,\Omega} = P(H_p + \sigma I)P = PLL^T P$, where the exact symmetric matrix factor L can be obtained by the exact Cholesky factorization with or without fill-reducing permutation. Then, a symmetric version of the preconditioned Hessian with the shifted Hessian preconditioner can be defined as

$$H_c = PL^{-1}P\frac{\partial^2 E(\phi)}{\partial \phi^2}PL^{-T}P. \tag{4.17}$$

Note that the preconditioned Hessian (4.17) is defined ideally, since we cannot afford the exact Cholesky decomposition for large problems in practice. Nevertheless, this ideal preconditioned Hessian helps us develop insight into the expected favorable behavior of the nonlinear PCG with our practical preconditioner near convergence.

Theorem 4.4.1. *The preconditioned Hessian operator with the ideal shifted Hessian preconditioner* (4.14) *given in* (4.17) *can be written in the following form*

$$L^{-1}H_pL^{-T} + WW_1^T + L^{-1}WW_2^T + L^{-1}H_pWW_3^T - h^dL^{-1}H_pL^{-T}WW^T, \tag{4.18}$$

for $\sigma > 0$ *and*

$$I + W(W_1 - h^dW)^T + L^{-1}WW_2^T + L^{-1}H_pWW_3^T, \tag{4.19}$$

for $\sigma = 0$. *Here,* W_1, W_2, $W_3 \in \mathbb{R}^{2n\times 2}$. *In other words, the ideal preconditioned Hessian is a rank-6 update of the identity matrix for* $\sigma = 0$, *and a rank-8 update of* $L^{-1}H_pL^{-T}$ *that is close to the identity matrix for a small* $\sigma > 0$.

Proof. The proof is given in the Appendix. □

Theorem 4.4.1 implies that almost all eigenvalues of the ideal preconditioned Hessian (4.17) are exactly or nearly 1, and there are only 6 or 8 eigenvalues that could be significantly different from 1. Among these 6 or 8 eigenvalues, there are 2 zero eigenvalues associated with the orthogonal projector P and have no impact on the convergence of the nonlinear PCG. Most importantly, such an observation of the spectrum is independent of the mesh size. Specifically, suppose $\sigma > 0$ and define the rank-8 matrix

$$R_8 = WW_1^T + L^{-1}WW_2^T + L^{-1}H_pWW_3^T - h^dL^{-1}H_pL^{-T}WW^T.$$

Then, by Theorem 4.4.1, we have

$$H_c = I + R_8 - \sigma L^{-1}L^{-T}. \tag{4.20}$$

Suppose α_i, ρ_i for $1 \le i \le 2n$ (satisfying $\alpha_i \le \alpha_{i+1}$, $\rho_i \le \rho_{i+1}$) are the eigenvalues of $I + R_8$ and $I + R_8 - \sigma L^{-1}L^{-T}$, respectively. Then, $2n-8$ eigenvalues among α_i's are 1 and at most 8 eigenvalues are not 1. Moreover, by the Bauer–Fike theorem [40], we have $|\alpha_i - \rho_i| \le |\sigma|\|L^{-1}L^{-T}\|_2$.

Also, we know that $L^{-1}L^{-T}$ and $(H_p + \sigma I)^{-1}$ have the same eigenvalues, since $H_p + \sigma I = LL^T$. Assume that the lowest positive eigenvalue of H_p (positive semidefinite) has an infimum $s^* > 0$ that is independent of the mesh size h, as $h \to 0$. It follows that the spectrum of $L^{-1}L^{-T}$ falls within $(0, \frac{1}{s^*+\sigma}]$, i.e., $\|L^{-1}L^{-T}\|_2 \le \frac{1}{s^*+\sigma}$. Then, we have

$$|\alpha_i - \rho_i| \le \frac{|\sigma|}{s^* + \sigma}, \tag{4.21}$$

which is guaranteed to be small if σ is small compared to s^*. In other words, for any $\sigma \ll s^*$, the eigenvalues of $I + R_8 - \sigma L^{-1}L^{-T}$ are not much different from those of $I + R_8$. This ensures that the preconditioned Hessian has a favorable eigenvalue distribution such that the nonlinear PCG with our ideal shifted Hessian preconditioner (where σ is sufficiently small) is expected to converge fairly quickly when approaching convergence. Note that with the incomplete Cholesky preconditioner obtained by a fixed drop tolerance, the condition number of $L^{-1}H_pL^{-T}$ deteriorates as the mesh size h decreases, so that nonlinear PCG needs more iterations to converge on a finer mesh.

4.5 Numerical experiments

In this section, we perform extensive experiments in 2D and 3D domains to validate our method. We compare our Hessian preconditioner with the Combined preconditioner proposed in [11]. In the following experiments, we consider the trapping potential: the harmonic plus quartic potential for $d = 2, 3$

$$V(\mathbf{x}) = (1-\alpha)(\gamma_x^2 x^2 + \gamma_y^2 y^2) + \frac{\kappa(x^2+y^2)^2}{4} + \begin{cases} 0, & d = 2, \\ \gamma_z^2 z^2, & d = 3. \end{cases} \tag{4.22}$$

Moreover, we take the initial wave function $\phi_{(0)}$ as the Thomas Fermi approximation [11, 16]

$$\phi_{(0)} = \frac{\phi^{TF}}{\|\phi^{TF}\|_{\ell^2}} \quad \text{with} \quad \phi^{TF}(\mathbf{x}) = \begin{cases} \sqrt{(\mu^{TF} - V(\mathbf{x}))/\eta}, & V(\mathbf{x}) < \mu^{TF} \\ 0, & \text{otherwise}, \end{cases} \tag{4.23}$$

where

$$\mu^{TF} = \frac{1}{2} \begin{cases} (4\eta\gamma_x\gamma_y)^{1/2}, & d = 2, \\ (15\eta\gamma_x\gamma_y\gamma_z)^{2/5}, & d = 3. \end{cases} \tag{4.24}$$

The stopping criterion we adopt is

$$\frac{|\, E(\phi_{n+1}) - E(\phi_n)\,|}{|\, E(\phi_n)\,|} \leq \epsilon = 10^{-14}, \tag{4.25}$$

Other stopping criterion and comparison between them can be found in [16]. In order to apply our Hessian preconditioner, we perform an inexact Cholesky factorization with the approximate minimal degree ordering and the drop tolerance is chosen to be 10^{-3} and $10^{-2.5}$ for experiments in 2D and 3D domains, respectively. We use the two stage preconditioning strategy. The Combined preconditioner is used at the first stage and our Hessian preconditioner is used at the second stage. We switch the preconditioner when $\frac{|E(\phi_{n+1})-E(\phi_n)|}{|E(\phi_n)|} \leq 10^{-7}$. After switching to the Hessian preconditioner, we update the Hessian preconditioner every 100 iterations experiments in 2D domains and 300 iterations in 3D domains.

4.5.1 Fast energy evaluation and line search methods

In this section, we perform several experiments to compare the performances of the nonlinear PCG with different line search methods (*with* or *without* fast evaluation of the energy) that we introduce in Section 4.3. Note that we cannot afford to implement the exact line search *without* the fast evaluation of the energy. Therefore, there are 5 schemes: (a) exact line search *with* fast evaluation; (b) quadratic line search *with* fast evaluation; (c) quadratic line search *without* fast evaluation; (d) backtracking line search *with* fast evaluation; (e) backtracking line search *without* fast evaluation. Note that the exact line search performed by MATLAB's built-in function `fminsearch` is parameter-free, and the quadratic and backtracking line search use a small number of parameters whose values are predetermined, independent of the test problems. Here, we test two cases: (I) $\eta = 100$, $\Omega = 0.9$ and $V(\mathbf{x})$ is chosen with $\gamma_x = \gamma_y = 1$, $\alpha = 0.5$ and $\kappa = 0$; (II) $\eta = 1000$, $\Omega = 2$ and $V(\mathbf{x})$ is chosen with $\gamma_x = \gamma_y = 1$, $\alpha = 1.2$ and $\kappa = 0.3$. The computational domain and mesh size are $\mathcal{D} = [-10, 10]^2$ and $h = \frac{1}{32}$, respectively. To make fair comparison, all the experiments are performed with the Combined preconditioner only. The results are summarized in Tables 4.5.1 and 4.5.1, respectively. We can see that the exact line search and quadratic line search are more competitive than the backtracking line search. Also, the exact line search could marginally improve the number of iterations compared with the quadratic line search. More importantly, the fast evaluation of the energy is always preferred.

Table 4.1: Comparison of line search for quadratic approximation and exact line search for case I.

$\eta = 100$, $\Omega = 0.9$	exact	quadratic		backtracking	
	fast	fast	slow	fast	slow
PCG iteration	141	142	133	205	205
time (sec)	24.45	25.53	43.1	35.67	60.37

Table 4.2: Comparison of line search for quadratic approximation and exact line search for case II.

$\eta = 1000$, $\Omega = 2$	exact	quadratic		backtracking	
	fast	fast	slow	fast	slow
PCG iteration	302	310	309	579	579
time (sec)	53.91	54.30	86.82	100.65	230.62

4.5.2 Partial spectrum of the preconditioned Hessian

In this section, we provide numerical examples to illustrate the partial spectrum of preconditioned Hessian with our Hessian preconditioner (4.17) at the converged ground state solution ϕ_c. Here, we have two cases: (I) $\eta = 500, \Omega = 0.8$, and $V(\mathbf{x})$ is chosen with $\gamma_x = \gamma_y = 1$, $\alpha = 0.5$ and $\kappa = 0$; (II) $\eta = 5000$ and $\Omega = 1$, and $V(\mathbf{x})$ is chosen with $\gamma_x = \gamma_y = 1$, $\alpha = 1.2$ and $\kappa = 0.3$. We use the MATLAB built-in function `eigs` to compute the partial spectrum for the preconditioned Hessian operators.

4.5.2.1 Partial spectrum with different shift σ

In this example, we apply the 8th order finite difference scheme to form (4.17), i.e, finite difference method for both the effective Hessian itself (4.13) and the Hessian preconditioner (4.14). Also, we perform an exact Cholesky factorization of the Hessian preconditioner for illustration, which is too expensive for large realistic problems. Here, we fix $h = \frac{1}{8}$ and the shift σ varies. The computational domain is $\mathcal{D} = [-10, 10]^2$. Tables 4.5.2.1 and 4.5.2.1 list the 10 smallest eigenvalues and 10 largest eigenvalues of the preconditioned Hessian operator for our Hessian preconditioner with different shifts σ. We can see that most of the eigenvalues of the precondition Hessian with Hessian preconditioner are approximately 1. These observations are consistent with Theorem 4.4.1.

Table 4.3: Partial spectrum of Preconditioned Hessian with different shift σ for Case I.

$\lambda_{\min}$	$\lambda_{\max}$	$\lambda_{\min}$	$\lambda_{\max}$	$\lambda_{\min}$	$\lambda_{\max}$
-6.49e-16	1.25	-4.96e-17	1.24	-4.49e-16	1.22
-1.66e-19	1.00	4.84e-17	1.00	-1.90e-16	1.00
2.33e-08	1.00	8.65e-08	1.00	1.62e-07	1.00
2.87e-05	1.00	3.00e-05	1.00	3.86e-05	1.00
5.18e-02	1.00	5.44e-03	1.00	5.53e-04	1.00
1.82e-01	1.00	2.18e-02	1.00	2.22e-03	1.00
5.05e-01	1.00	9.28e-02	1.00	1.01e-02	1.00
5.43e-01	1.00	1.06e-01	1.00	1.17e-02	1.00
8.24e-01	1.00	3.20e-01	1.00	4.49e-02	1.00
8.49e-01	1.00	3.61e-01	1.00	5.35e-02	1.00
(a) $\sigma = 10^{-3}$		(b) $\sigma = 10^{-2}$		(c) $\sigma = 10^{-1}$	

Table 4.4: Partial spectrum of Preconditioned Hessian with different shift σ for Case II.

$\lambda_{\min}$	$\lambda_{\max}$	$\lambda_{\min}$	$\lambda_{\max}$	$\lambda_{\min}$	$\lambda_{\max}$
-3.69e-16	1.77	-2.96e-16	1.17	-6.90e-17	1.16
2.92e-16	1.17	-8.12e-17	1.02	-4.16e-17	1.00
3.11e-06	1.00	4.33e-07	1.00	9.30e-06	1.00
4.55e-05	1.00	4.71e-05	1.00	4.65e-05	1.00
9.60e-01	1.00	6.97e-01	1.00	1.93e-01	1.00
9.60e-01	1.00	7.11e-01	1.00	1.94e-01	1.00
9.83e-01	1.00	8.53e-01	1.00	3.75e-01	1.00
9.85e-01	1.00	8.66e-01	1.00	4.03e-01	1.00
9.86e-01	1.00	8.90e-01	1.00	4.04e-01	1.00
9.89e-01	1.00	9.12e-01	1.00	4.90e-01	1.00
(a) $\sigma = 10^{-3}$		(b) $\sigma = 10^{-2}$		(c) $\sigma = 10^{-1}$	

4.5.2.2 Partial spectrum with different preconditioners

In this example, we compare the partial spectrum of preconditioned Hessian with the state-of-the-art Combined preconditioner [11] and our Hessian preconditioner. To be consistent with [11], we compute the partial spectrum based on the the expression of the non-symmetric preconditioned Hessian (4.16). The Hessian operator (4.13) is discretized in Fourier pseudo-spectral scheme for both cases. Moreover, we apply the Combined preconditioner in Fourier pseudo-spectral scheme and the Hessian preconditioner in 8th order finite difference scheme, which is consistent with the preconditioning strategy proposed in Section 4.2. To make fair comparison, we scale the computed eigenvalues so that the largest eigenvalue of both preconditioned Hessian is of the same value for each experiment. Also, we perform an inexact Cholesky factorization with drop tolerance 10^{-3} of the shifted Hessian preconditioner (4.14) with shift $\sigma = 10^{-3}$, permutated by the approximate minimal degree ordering. For Case I, we fix $h = \frac{1}{16}$ and vary the domain length L from 4 to 12. For Case II, we fix $L = 10$ and vary h from $\frac{1}{4}$ to $\frac{1}{16}$. Tables 4.5.2.2 and 4.5.2.2 list the 10 smallest scaled eigenvalues and 10 largest scaled eigenvalues of the preconditioned Hessian operator (4.16) for the Combined preconditioner and the Hessian preconditioner with different domain $\mathcal{D}$, respectively. Furthermore, Tables 4.5.2.2 and 4.5.2.2 list the 10 smallest scaled eigenvalues and 10 largest scaled eigenvalues of the preconditioned Hessian operator for the Combined preconditioner and Hessian preconditioner with different mesh size h, respectively. From these results, we can see that the conditioning deteriorates for both preconditioners as both the spatial resolution and the size of the domain increase, which is consistent with the observation in [11]. However, the preconditioned Hes-

sian with Hessian preconditioner has a more favorable eigenvalue distribution and smaller condition number compared with the Combined preconditioner. More importantly, the nonlinear PCG with the Hessian preconditioner is expected to converge fairly quickly when approaching convergence.

Table 4.5: Partial spectrum of Preconditioned Hessian with Combined preconditioner for Case I.

$\lambda_{\min}$	$\lambda_{\max}$	$\lambda_{\min}$	$\lambda_{\max}$	$\lambda_{\min}$	$\lambda_{\max}$
-1.02e-16	1.00	7.38e-16	1.00	-1.23e-16	1.00
5.09e-17	1.00	3.77e-16	1.00	1.68e-16	1.00
1.25e-03	1.00	1.31e-09	0.98	2.56e-11	0.98
2.18e-03	1.00	3.71e-05	0.98	3.71e-05	0.98
2.70e-03	1.00	3.71e-05	0.98	3.71e-05	0.98
3.48e-03	1.00	2.26e-04	0.98	2.26e-04	0.98
5.18e-03	1.00	3.40e-04	0.98	2.26e-04	0.98
7.27e-03	1.00	4.62e-04	0.98	3.39e-04	0.98
8.22e-03	0.98	7.11e-04	0.98	3.39e-04	0.98
1.06e-02	0.98	7.28e-04	0.98	4.62e-04	0.98
(a) $\mathcal{D} = [-4, 4]^2$		(b) $\mathcal{D} = [-8, 8]^2$		(c) $\mathcal{D} = [-12, 12]^2$	

Table 4.6: Partial spectrum of Preconditioned Hessian with incomplete Cholesky Hessian preconditioner for Case I.

$\lambda_{\min}$	$\lambda_{\max}$	$\lambda_{\min}$	$\lambda_{\max}$	$\lambda_{\min}$	$\lambda_{\max}$
-2.91e-16	1.00	-1.96e-15	1.00	-1.26e-16	1.00
1.07e-16	1.00	1.40e-16	1.00	1.34e-15	1.00
6.85e-03	1.00	7.02e-09	1.00	1.30e-10	1.00
1.13e-02	1.00	2.05e-04	1.00	1.97e-04	1.00
1.26e-02	1.00	2.23e-04	1.00	2.02e-04	1.00
1.69e-02	1.00	1.33e-03	1.00	1.23e-03	1.00
2.78e-02	1.00	1.38e-03	1.00	1.31e-03	1.00
3.62e-02	1.00	1.80e-03	1.00	1.69e-03	1.00
3.96e-02	1.00	1.92e-03	1.00	1.77e-03	1.00
5.81e-02	1.00	2.77e-03	1.00	2.58e-03	1.00
(a) $\mathcal{D} = [-4, 4]^2$		(b) $\mathcal{D} = [-8, 8]^2$		(c) $\mathcal{D} = [-12, 12]^2$	

4.5.3 Numerical experiments in 2D

In this section, we apply our method to compute the ground state for some 2D BECs problems with strong repulsive interaction and rotational speed (large values of η and Ω), which are more relevant for real physical problems. We compare our Hessian preconditioner with the state-of-the-art Combined preconditioner. The maximum iteration number is set to be 100000. All the

Table 4.7: Partial spectrum of Preconditioned Hessian with Combined preconditioner for Case II.

$\lambda_{\min}$	$\lambda_{\max}$	$\lambda_{\min}$	$\lambda_{\max}$	$\lambda_{\min}$	$\lambda_{\max}$
-2.66e-17	1.00	-1.38e-17	1.00	-5.37e-17	1
-1.40e-17	1.00	1.99e-17	1.00	-1.55e-17	1
3.58e-04	1.00	1.76e-04	0.99	5.82e-08	0.99
3.58e-04	1.00	1.76e-04	0.99	3.56e-07	0.99
9.42e-04	1.00	4.63e-04	0.98	1.88e-05	0.98
9.75e-04	1.00	5.02e-04	0.98	1.99e-05	0.98
1.40e-03	1.00	5.77e-04	0.98	7.17e-05	0.98
1.78e-03	1.00	8.26e-04	0.98	7.93e-05	0.98
1.78e-03	0.98	8.26e-04	0.98	7.95e-05	0.98
2.75e-03	0.98	9.92e-04	0.98	7.96e-05	0.98
(a) $h = \frac{1}{4}$		(b) $h = \frac{1}{8}$		(c) $h = \frac{1}{16}$	

Table 4.8: Partial spectrum of Preconditioned Hessian with incomplete Cholesky Hessian preconditioner for Case II.

$\lambda_{\min}$	$\lambda_{\max}$	$\lambda_{\min}$	$\lambda_{\max}$	$\lambda_{\min}$	$\lambda_{\max}$
1.29e-17	1.00	-3.45e-17	1.00	-1.79e-16	1.00
2.76e-17	0.98	2.32e-16	1.00	2.00e-16	1.00
1.21e-01	0.96	1.43e-02	1.00	1.23e-06	1.00
1.30e-01	0.96	1.48e-02	1.00	7.51e-06	1.00
2.28e-01	0.96	3.87e-02	1.00	3.58e-04	1.00
2.28e-01	0.96	4.04e-02	1.00	3.79e-04	1.00
2.31e-01	0.96	6.10e-02	1.00	1.55e-03	1.00
2.31e-01	0.96	6.95e-02	1.00	1.58e-03	1.00
2.31e-01	0.96	7.05e-02	0.98	1.59e-03	1.00
2.31e-01	0.96	8.66e-02	0.98	1.71e-03	1.00
(a) $h = \frac{1}{4}$		(b) $h = \frac{1}{8}$		(c) $h = \frac{1}{16}$	

experiments are performed on a Ubuntu 22.04 LTS (64 bit) PC-Intel(R) Core(TM) i7-4700 CPU 2.40 GHz, 32 GB of DDR3 1600MHz RAM running MATLAB R2022b. Our numerical results show that the Hessian preconditioner is more efficient than the Combined preconditioner especially for the fast rotational BEC problems.

Firstly, $V(\mathbf{x})$ is chosen with $\gamma_x = \gamma_y = 1$, $\alpha = 1.2$ and $\kappa = 0.3$ [11]. The computational domain and mesh sizes are $\mathcal{D} = [-20, 20]^2$ and $h = \frac{1}{32}$. We compute the ground states ϕ_g of rotating BECs with large values of η and Ω. In Table 4.9, η is fixed to be 10000 and Ω is chosen from 1 to 5. In Table 4.10, Ω is fixed to be 5 and η is chosen from 1000 to 20000. Tables 4.9 and 4.10 lists the iterations, runtime and final energy functional our method attain with Combined preconditioner and Hessian preconditioner, respectively. Also, the lower final energy value for the same (η, Ω) is highlighted in bold. The contour plot of the density function $|\phi_g(\mathbf{x})|^2$ with the Combined preconditioner and the Hessian preconditioner are shown in Figure 4.1 and 4.2, respectively. For example, in Table 4.9, when $\eta = 10000, \Omega = 5$, the nonlinear PCG *only* with the Combined preconditioner takes 26522 iterations to attain -485.0282069197 in 21126.00 seconds, whereas, only 4488 iterations is required to attain -485.0305526536 in 4681.19 seconds with the Hessian preconditioner. Tables 4.9 and 4.10 show the advantage of our Hessian preconditioner involving Ω over the Combined preconditioner that disregards Ω. We can see that with larger values of nonlinearity η and rotating speed Ω, our Hessian preconditioner gains more advantage in runtime. Also, the nonlinear PCG with our Hessian preconditioner tends to achieve a lower final energy.

Table 4.9: Performance comparison of PCG with two preconditioners for $\eta = 10000$ and different Ω values.

Ω	PCG iteration		time(sec)		final $E_{\eta,\Omega}$	
	Combined	Hessian	Combined	Hessian	Combined	Hessian
1	724	2088	576.51	2052.01	63.02007542539	**62.96553732649**
1.5	749	697	593.38	583.06	53.26795985753	**53.26795985751**
2	4929	2443	3885.88	2399.98	37.59961999660	**37.59961999657**
2.5	5770	3287	4589.90	3137.76	13.63739471900	**13.63739471896**
3	16435	6226	12885.70	6347.01	**−23.48312229660**	-23.48295831441
3.5	8653	3612	6895.37	3733.51	-82.54564206625	**−82.54564207131**
4	25890	6047	20546.26	6430.85	-172.7171085876	**−172.7188268092**
4.5	18115	3701	14125.19	3868.01	-303.3183033037	**−303.3185838060**
5	26522	4488	21126.00	4681.19	-485.0282069197	**−485.0305526536**

Next, we compare the performance of our Hessian preconditioner with the Combined preconditioner to solve some more difficult problems. Here, $V(\mathbf{x})$ is chosen with $\gamma_x = 10$, $\gamma_y = 1$, $\alpha = 2$

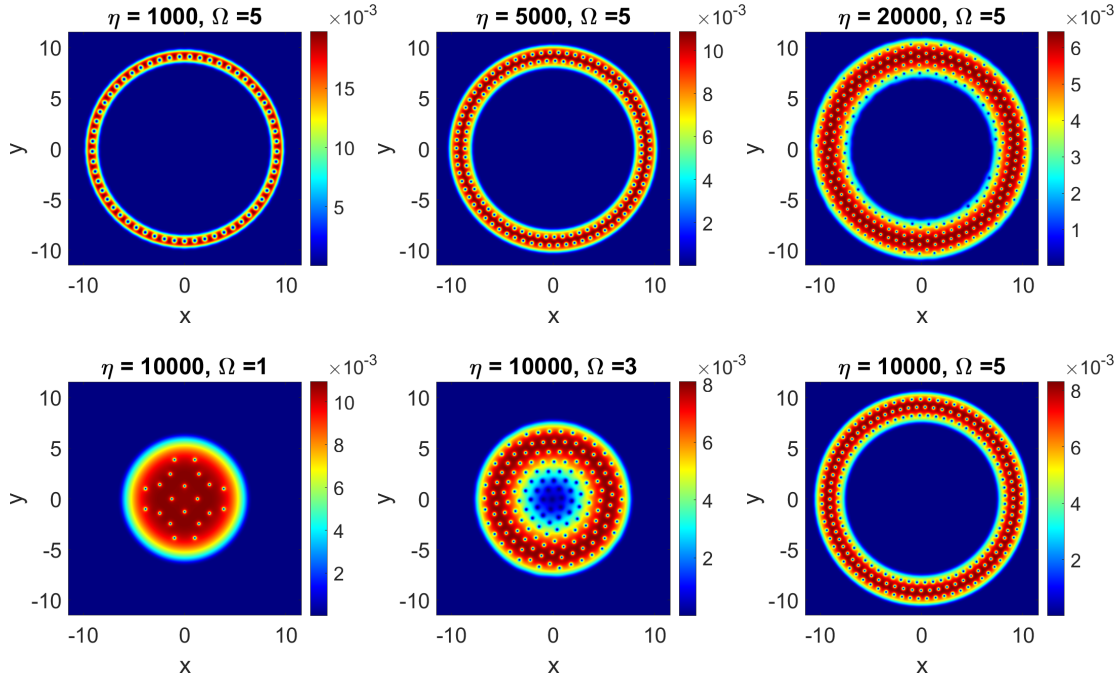


Figure 4.1: Corresponding contour plots of the density function with the Combined preconditioner $|\phi_g(\mathbf{x})|^2$ in Table 4.9 and 4.10.

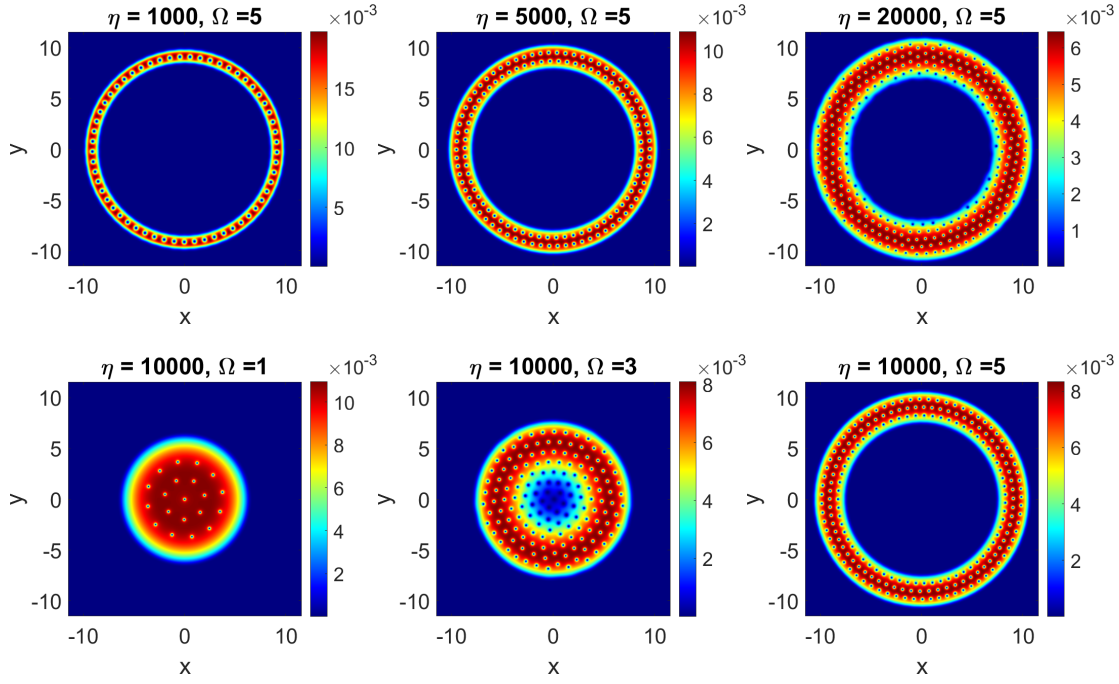


Figure 4.2: Corresponding contour plots of the density function with the Hessian preconditioner $|\phi_g(\mathbf{x})|^2$ in Table 4.9 and 4.10.

Table 4.10: Performance comparison of PCG with two preconditioners for $\Omega = 5$ and different η values.

η	PCG iteration		time(sec)		final $E_{\eta,\Omega}$	
	Combined	Hessian	Combined	Hessian	Combined	Hessian
1000	12922	2426	10322.28	2446.36	-522.1631296805	$-$**522.1631296901**
2000	10846	2164	8727.81	2192.61	-516.1164313740	$-$**516.1164313839**
5000	17673	5349	14009.90	5710.81	-502.4144226059	$-$**502.4145222867**
10000	26522	4488	21126.00	4681.19	-485.0282069197	$-$**485.0305526536**
20000	60757	6401	47431.78	6956.86	$-$**457.6996232199**	-457.6981668537

and $\kappa = 3$. We fix $\eta = 25000$ and vary Ω from 4 to 16. We take $L_x = L_y = 13$, $h = 1/64$. The results are showed in Table 4.11. Figure 4.3 and 4.4 show the contour plot of the density function $|\phi_g(\mathbf{x})|^2$ with the Combined preconditioner and the Hessian preconditioner, respectively. Note that the nonlinear PCG with the Combined preconditioner does not terminate after 10000 iterations, thus we record the energy and the runtime it attains at the 10000 iterations. More importantly, Table 4.11 shows that our Hessian preconditioner gains significant advantage over the Combined preconditioner.

Table 4.11: Performance comparison of PCG with two preconditioners for $\eta = 25000$ and different Ω values.

Ω	PCG iteration		time(sec)		final $E_{\eta,\Omega}$	
	Combined	Hessian	Combined	Hessian	Combined	Hessian
4	51864	17454	74290.97	33295.91	141.3951364011e	**141.3951364033**
8	100000+	39510	143848.1+	83809.49	-294.0500897923	$-$**294.0521455922**
12	98901	9616	170704.1	20936.99	$-$**1871.149053296**	-1871.148855358
16	100000+	30003	168655.4+	68113.02	-5913.255005955	$-$**5913.256644684**

4.5.4 Numerical experiments in 3D

In this section, we apply our method to compute some 3D problems. We perform the 3D experiments on a single node with 16 cores on Clemson Palmetto Cluster running MATLAB R2022a. We test four cases : (i) $\eta = 15000$, $\Omega = 4$; (ii) $\eta = 15000$, $\Omega = 5$; (iii) $\eta = 25000$, $\Omega = 4$; (iv) $\eta = 25000$, $\Omega = 6$. The mesh size is $h = \frac{1}{16}$ for all cases. For (i) and (ii), $V(\mathbf{x})$ is chosen with $\gamma_x = \gamma_y = 1, \gamma_z = 1, \alpha = 0.3$ and $\kappa = 1.4$. The computational domain is $D = [-15, 15]^2 \times [-8, 8]$. For (iii) and (iv), $V(\mathbf{x})$ is chosen with $\gamma_x = \gamma_y = 1, \gamma_z = 3, \alpha = 0.3$ and $\kappa = 1.4$. The computational domain is $D = [-10, 10]^2 \times [-5, 5]$. We summarize the results in Table 4.12 and Table 4.13. Figure 4.5 and 4.6 show the isosurfaces $|\phi_g(\mathbf{x})|^2 = 10^{-3}$ and surface plots of $|\phi_g(x, y, z = 0)|^2$ for all the cases.

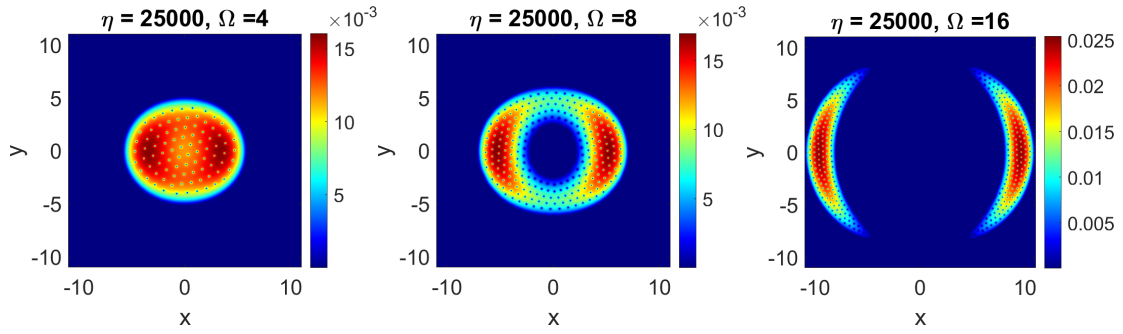


Figure 4.3: Corresponding contour plots of the density function with the Combined preconditioner $|\phi_g(\mathbf{x})|^2$ in Table 4.11.

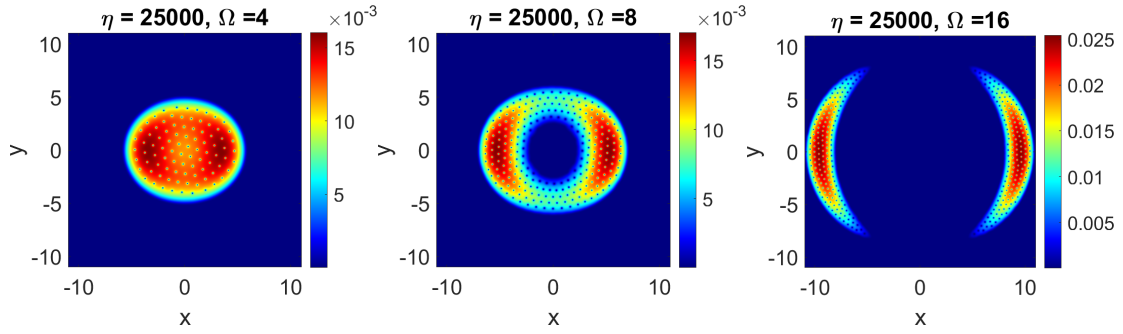


Figure 4.4: Corresponding contour plots of the density function with the Hessian preconditioner $|\phi_g(\mathbf{x})|^2$ in Table 4.11..

From these results, we can see that our method work efficiently tackling challenging problems and our Hessian preconditioner is still competitive compared with the Combined preconditioner.

Table 4.12: Performance comparison of PCG with two preconditioners for Cases I and II.

$(\eta,\ \Omega)$	PCG iteration		time(sec)		final $E_{\eta,\Omega}$	
	Combined	Hessian	Combined	Hessian	Combined	Hessian
(15000, 4)	14568	3864	378007.4	156913.9	-210.8746065833	$\mathbf{-210.8746066226}$
(15000, 5)	28023	10866	691078.8	448749.6	-529.2941298728	$\mathbf{-529.2943293465}$

Table 4.13: Performance comparison of PCG with two preconditioners for Cases III and IV.

$(\eta,\ \Omega)$	PCG iteration		time(sec)		final $E_{\eta,\Omega}$	
	Combined	Hessian	Combined	Hessian	Combined	Hessian
(25000, 4)	3509	2325	29258.16	23823.73	75.88162274531	**75.88162274514**
(25000, 6)	16929	7611	140570.9	74431.28	1.258275896279	**1.258275895894**

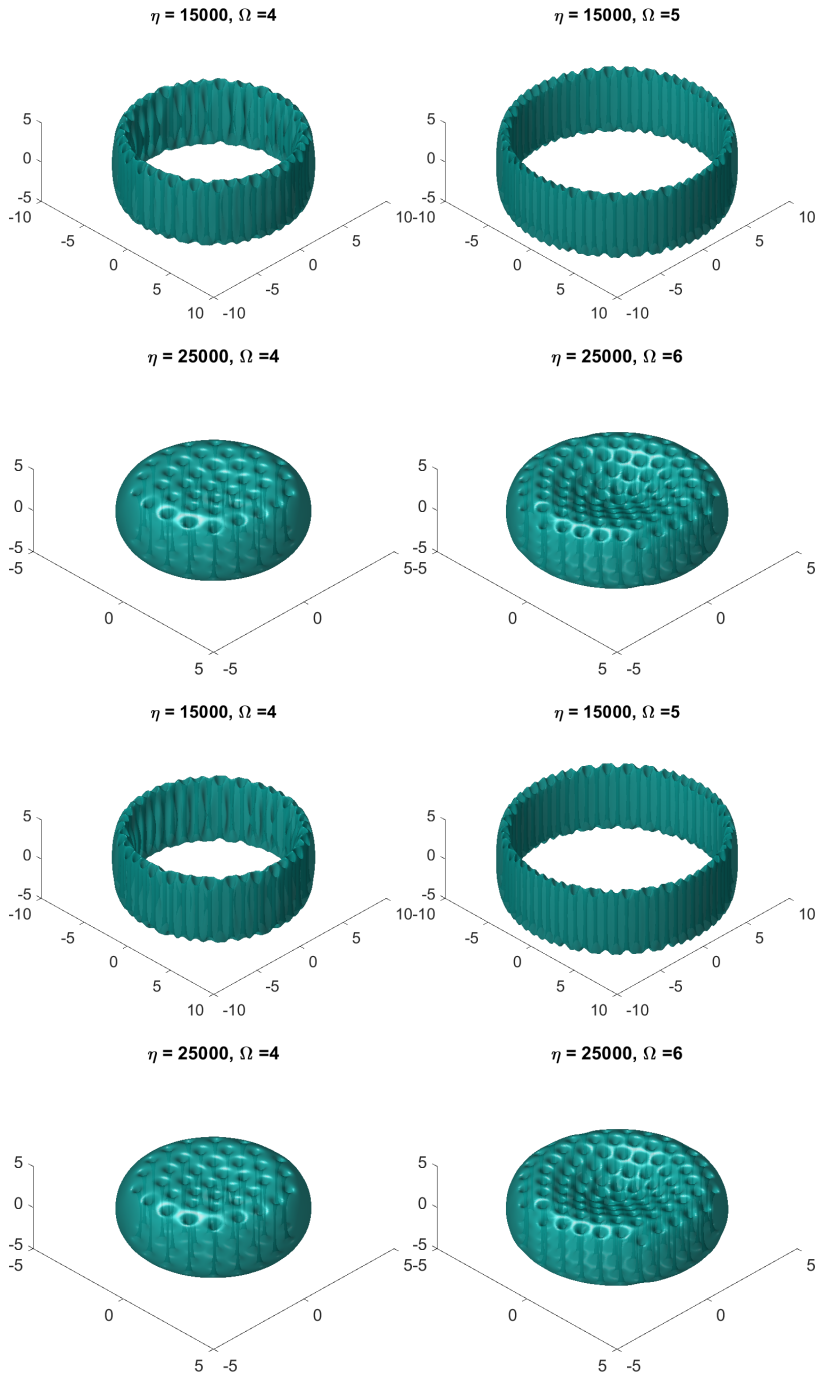


Figure 4.5: Corresponding isosurface $|\phi_g(\mathbf{x})|^2 = 10^{-3}$ in Table 4.12 (Top 4 : Combined; Lower 4: Hessian).

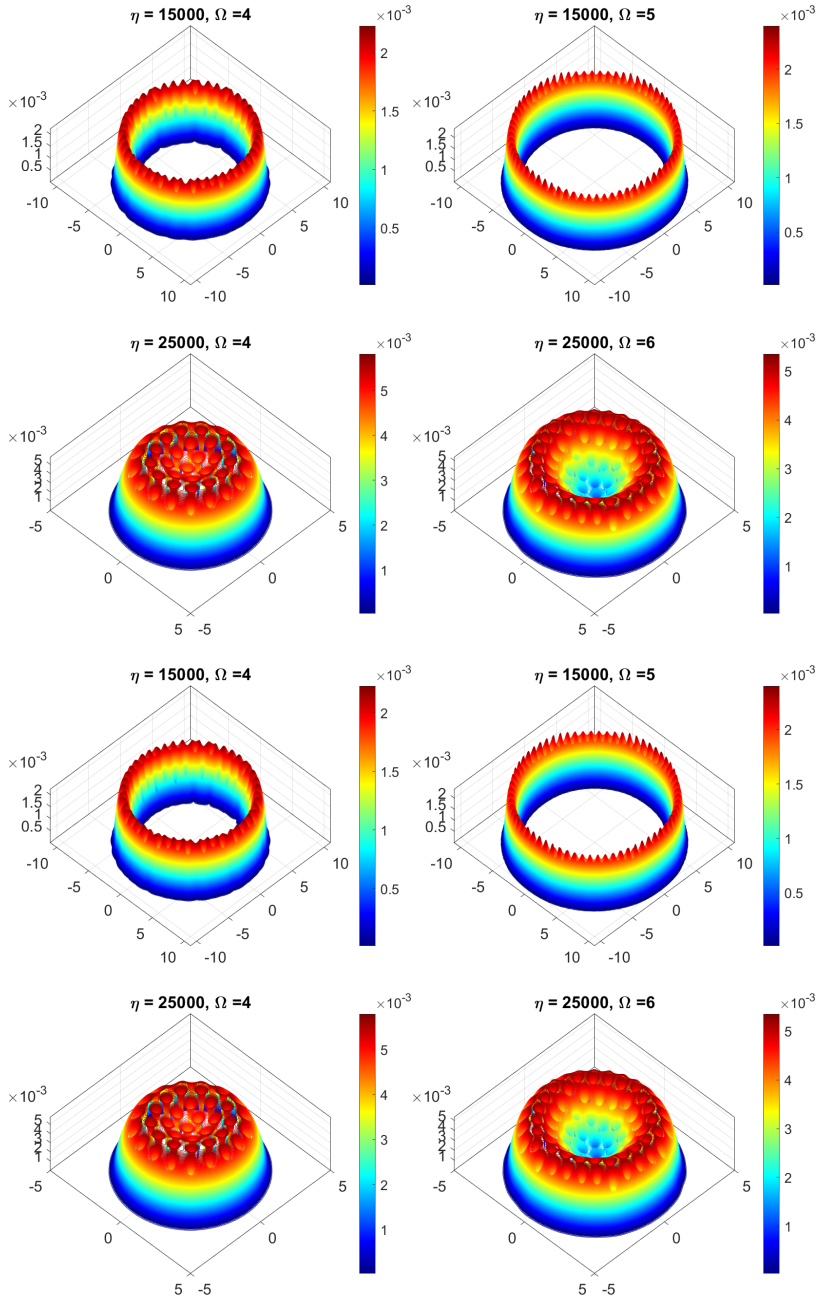


Figure 4.6: Corresponding surface plot of $|\phi_g(x, y, z = 0)|^2$ in Table 4.12 (Top 4: Combined; Lower 4: Hessian).

Chapter 5

The Bethe-Salpeter eigenvalue problem

In this chapter, we propose an extension of Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) [63] that involves polynomial filtered Ritz vectors and preconditioned eigenresidual vectors simultaneously. This new algorithm usually exhibits rapid and robust convergence without the dependence on a polynomial filter or a preconditioner of very high quality. The projection methods used in this chapter are geared toward a limited memory implementation for computing a potentially large number (e.g., up to 200) of the smallest eigenvalues of (1.25) or (1.32). We construct an m-step ideal version of our proposed method that also involves the polynomial filtered Ritz vectors and preconditioned eigenresidual vectors simultaneously, and establish an asymptotic global quasi-optimality of the ideal method, which is similar to the global optimality of the conjugate gradient method for the iterative solution of a symmetric positive definite linear system. We provide comprehensive numerical experiments to validate our algorithms and demonstrate their competitiveness against a few widely used iterative methods for solving the BSE eigenproblems.

The remainder of this chapter is organized as follows. We provide a description of the existing algorithms in Section 5.1. In Section 5.2, we present a detailed description of our method. In Section 5.3 , we construct an ideal variant of our new locally optimal method and develop an analysis of its asymptotic global quasi-optimality. Section 5.4 provides numerical results to validate the algorithms.

5.1 Review of existing algorithms

In this section, we review two classes of non-Krylov subspace iterative methods for computing the lowest partial spectrum of the product eigenproblem (1.22). Our new method is based on a proper combination of the strengths of both algorithms.

5.1.1 *M*-LOBPCG

LOBPCG is widely used for computing extreme eigenvalues of large sparse symmetric matrices. The so-called M-LOBPCG is similar to the LOBPCG method applied to a symmetric matrix, where the standard inner product is replaced with the M-inner product.

Starting from an initial approximation of the desired eigenvectors $X^{(0)} \in \mathbb{R}^{n\times k}$, the search subspace $\mathcal{S}^{(i)}$ at the ith iteration constructed by M-LOBPCG is

$$\mathcal{S}^{(i)} = \text{span}\{X^{(i)}, T^{-1}R^{(i)}, P^{(i)}\}, \tag{5.1}$$

where $X^{(i)}$ is the eigenvector approximation at the ith iteration, $T^{-1}R^{(i)} \in \mathbb{R}^{n\times k}$ is the preconditioned eigenresidual vectors, and the conjugate search direction $P^{(i)} \in \mathbb{R}^{n\times k}$ is defined as

$$P^{(i)} = X^{(i)} - X^{(i-1)}C_X, \tag{5.2}$$

where $C_X \in \mathbb{R}^{k\times k}$ are determined by the *primitive* Ritz vectors from the Rayleigh-Ritz procedure of last iteration. In addition, we observed experimentally that the search direction $P^{(i)}$ of M-LOBPCG could also be constructed by the linear least squares, namely, $P^{(i)} = X^{(i)} - X^{(i-1)}C_X$, where $C_X = X^{(i-1)\dagger}X^{(i)}$ to minimize the 2-norm of each column of $P^{(i)}$. However, this will take more runtime than (5.2) to obtain the coefficient matrix C_X. Mathematically, these two ways to extract coefficient matrix C_X construct the same search subspace $\mathcal{S}^{(i)}$ at the ith iteration such that

$$\mathcal{S}^{(i)} = \text{span}\{X^{(i)}, T^{-1}R^{(i)}, X^{(i-1)}\}. \tag{5.3}$$

At each step, we orthonormalize the the basis vectors of $\mathcal{S}^{(i)}$, denoted as $S^{(i)}$, so that $\langle S^{(i)}, S^{(i)}\rangle_M = I$. The Rayleigh-Ritz projection of KM with respect to the M-inner product onto

the subspace spanned by $S^{(i)}$ gives the projected eigenvalue problem

$$S^{(i)T}MKMS^{(i)}C = C\Theta^2, C^TC = I, \tag{5.4}$$

where Θ^2 is a $k \times k$ diagonal matrix which contains the k smallest eigenvalues of $S^{(i)T}MKMS^{(i)}$, whose associated eigenvectors are given by the orthonormal columns of the matrix $C \in \mathbb{R}^{3k\times k}$. Then, the approximations to the targeted eigenvectors of KM are given by the Ritz vectors $S^{(i)}C$, and the targeted eigenvalues are approximated by the diagonal entries of Θ^2. In short, in order to solve the product eigenvalue problem (1.22) , the M-LOBPCG method requires the projected subspace $S^{(i)}$ to stay M-orthonormal and the projected eigenvalue problem becomes the one for $S^{(i)T}MKMS^{(i)}$. Details of an implementation of M-LOBPCG are provided in [98].

5.1.2 Chebyshev M-Davidson

To achieve fast convergence, the choice of preconditioner in M-LOBPCG plays an important role. If no preconditioner of good quality is available, the polynomial filtering technique can be an effective alternative. In [110, 111], the Chebyshev Davidson method was proposed to employ a shifted and scaled Chebyshev polynomial to magnify the components of the desired eigenvectors and suppress those of the undesired ones.

For a symmetric matrix $A \in \mathbb{R}^{n\times n}$, suppose the spectrum of A falls within $[a_0, b]$ and the desired eigenvalues are in the interval $[a_0, a]$, where $a_0 < a < b$. With the linear transform $\phi(t) = \frac{a+b-2t}{b-a}$, the decreasing shifted and scaled Chebyshev polynomial filter is constructed as

$$F_d(t) = \frac{T_d(\phi(t))}{T_d(\phi(a_0))}, \tag{5.5}$$

where $T_d(t)$ is the d-th real Chebyshev polynomial of the first kind. It is easy to verify that $F_d(a_0) = 1$ and $F_d(t) \to 0$ exponentially with d for any $t \in [a, b]$. Therefore, for an eigenvector approximation v, $F_d(A)v$ amplifies the components of v in the directions of the eigenvectors associated with eigenvalues in $[a_0, a]$ and suppresses those in the directions of the eigenvectors associated with eigenvalues in $[a, b]$.

Specifically, to enable the effect of eigenvalue filtering, define $\sigma_i = \frac{T_{i-1}(\phi(a_0))}{T_i(\phi(a_0))}$. Through the

three-term recurrence relation between Chebyshev polynomials, we have

$$\sigma_1 = \frac{\omega}{\gamma - a_0}, \quad \sigma_i = \left(\frac{2}{\sigma_1} - \sigma_{i-1}\right)^{-1} \quad \text{for} \quad i = 2, 3, \cdots, d. \tag{5.6}$$

where $\gamma = \frac{a+b}{2}$ and $\omega = \frac{b-a}{2}$. The shifted and scaled polynomial could also be obtained in a recursive manner

$$F_i(t) = 2\sigma_i \frac{\gamma - t}{\omega} F_{i-1}(t) - \sigma_{i-1}\sigma_i F_{i-2}(t) \quad \text{for} \quad i = 2, 3, \cdots, d. \tag{5.7}$$

where $F_0(t) = 1$ and $F_1(t) = \frac{(\gamma - t)\sigma_1}{\omega}$.

Algorithm 4 displays a detailed description of the block version of the shifted and scaled Chebyshev polynomial filter for KM, which computes $X_F = F_d(KM)X$.

Algorithm 4 Chebyshev Filtered Iteration

Input: SPD matrices $K, M \in \mathbb{R}^{n\times n}$, eigenvector approximations $X \in \mathbb{R}^{n\times k}$, and parameters d, a, b, a_0 for Chebyshev polynomial filter F_d.
Output: $X_F = F_d(KM)X$

```
ω = (b−a)/2, γ = (b+a)/2, σ = ω/(γ−a_0), τ = 2/σ
X̂ = (γX − K(MX))σ/ω
for i = 2 : d do
    σ̂ = 1/(τ − σ)
    X_F = 2(γX̂ − K(MX̂))σ̂/ω − σσ̂X
    X = X̂, X̂ = X_F, σ = σ̂
end for
```

The core of Chebyshev Davidson is to expand the search subspace with polynomial filtered Ritz vectors. Similarly, the idea of Chebyshev M-Davidson is rather simple. Instead of using the preconditioned gradient (eigenresidual vectors), we use the polynomial filtered Ritz vectors to expand the search subspace, that is

$$\mathcal{S}^{(i)} = \text{span}\{S^{(i-1)}, X_F^{(i)}\}, \tag{5.8}$$

where $X_F^{(i)} = F_d^{(i)}(KM)X^{(i)}$, $X^{(i)}$ is the block of Ritz vectors obtained from Rayleigh-Ritz based on $S^{(i-1)}$ and $F_d^{(i)}(KM)$ is the polynomial filter adopted at the ith step. It is necessary to keep basis vectors of the projected subspace $S^{(i)}$ to be M-orthonormal to solve the product eigenvalue problem (1.22) with Chebyshev M-Davidson. We should point out that the Chebyshev M-Davidson method is equivalent to the Chebyshev Davidson method for the LREP proposed in [92]. An outline

of unrestarted Chebyshev M-Davidson is given in Algorithm 5. For difficult problems, they require too many iterations to converge, resulting in impractical memory and computational demands. A restart is needed when the projected subspace $S^{(i)}$ grows large.

Algorithm 5 Chebyshev M-Davidson

Input: SPD matrices $K, M \in \mathbb{R}^{n\times n}$, initial eigenvector approximations $X^{(0)} \in \mathbb{R}^{n\times k}$, Chebyshev polynomial filter F_d.

Output: Eigenpair approximations (Θ^2, X) satisfy $KMX = X\Theta^2$.

1: $X \leftarrow X^{(0)}$, $X_F \leftarrow F_d(KM)X^{(0)}$
2: $S \leftarrow [X, X_F]$
3: **while** convergence not reached **do**
4: $S \leftarrow M\text{-orth}\{S\}$[1]
5: Solve the projected eigenvalue problem $S^TMKMSC = C\Theta^2$ to obtain the desired lowest k eigenvalue and eigenvector approximations (Θ^2, X), where $X = SC \in \mathbb{R}^{n\times k}$ and $\Theta \in \mathbb{R}^{k\times k}$ is diagonal
6: Update the polynomial filter F_d
7: $X_F \leftarrow F_d(KM)X$
8: $S \leftarrow [S, X_F]$
9: **end while**

5.2 The Chebyshev M-LOBPCG method

In this section, we propose our new algorithm for computing the lowest eigenvalues of the product eigenvalue problems (1.25) and (1.32). This method, referred to as Chebyshev M-LOBPCG, can be considered as a balanced combination of Chebyshev M-Davidson and M-LOBPCG.

5.2.1 Motivation

To achieve fast convergence for the M-LOBPCG and the Chebyshev M-Davidson algorithms, the choice of the preconditioner and polynomial filter plays a crucial role. For large problems, as exact matrix factorizations are prohibitive or infeasible, we may rely on preconditioners such as the incomplete Cholesky factorization. An inexact Cholesky factorization with a lower drop tolerance could lead to faster convergence in terms of iterations, however, it cannot guarantee less runtime. Similarly, one challenge of the polynomial filter technique is to select the "optimal" degree of the polynomial. Higher degrees may lead to an excessive number of matrix-vector multiplications and lower degrees may result in very slow convergence. Also, if the desired interval $[a_0, a]$ is very narrow compared to the undesired interval $[a, b]$, it requires a very high degree to achieve a rapid

[1] $M\text{-orth}\{S\}$ returns an M-orthonormal basis of the subspace spanned by the columns of S.

rate of convergence, which leads to expensive computational cost. For non-expert users, it might be hard to select an appropriate preconditioner or polynomial filter. It is necessary to develop a black-box algorithm that does not heavily depend on users' choice of preconditioners or polynomial filters.

The motivation of our proposed Chebyshev M-LOBPCG is to cater to the needs for such a black-box algorithm. In order to lower the demand for the quality of preconditioners and polynomial filters in M-LOBPCG and Chebyshev M-Davidson, respectively, we propose to make use of both the preconditioned gradient and polynomial filtered Ritz vectors together into the projected subspace. To the best of our knowledge, no study has been conducted for solving matrix eigenvalue problems based on a combination of the preconditioned gradient and polynomial filtered Ritz vectors.

5.2.2 Outline of the Method

Our proposed Chebyshev M-LOBPCG can be considered as a special variant of M-LOBPCG. We keep the three-term recurrence structure and apply the preconditioner to the eigenresidual vectors. Different from M-LOBPCG, at each step before the Rayleigh-Ritz procedure, we replace the Ritz vectors with the polynomial filtered Ritz vectors from the last iteration in the projected subspace. Usually, this technique provides a projected subspace containing more accurate eigenvector approximation compared with M-LOBPCG.

Starting from an initial approximation of the desired eigenvectors $X^{(0)} \in \mathbb{R}^{n\times k}$, the search subspace $\mathcal{S}^{(i)}$ at the ith iteration is

$$\mathcal{S}^{(i)} = \text{span}\{X_F^{(i)}, W^{(i)}, P^{(i)}\} \tag{5.9}$$

where $X^{(i)}$ is the initial desired eigenvector approximation obtained in the ith iteration by the Rayleigh-Ritz projection onto $\mathcal{S}^{i-1}$, $X_F^{(i)}$ is the polynomial filtered outcome of $X^{(i)}$, i.e., $X_F^{(i)} = F_d^{(i)}(KM)X^{(i)}$, and $W^{(i)} = T^{-1}R^{(i)}$ is the preconditioned eigenresidual vectors for $X_F^{(i)}$. Similar to M-LOBPCG, the block of search direction $P^{(i)}$ is defined as

$$P^{(i)} = X_F^{(i)} - X_F^{(i-1)}C_X, \tag{5.10}$$

where C_X is obtained by minimizing the 2-norm of each column of $X_F^{(i)} - X_F^{(i-1)} C_X$, namely,

$$C_X = X_F^{(i-1)\dagger} X_F^{(i)}. \tag{5.11}$$

Therefore, we have

$$\mathcal{S}^{(i)} = \text{span}\{X_F^{(i)}, W^{(i)}, X_F^{(i-1)}\}, \tag{5.12}$$

which is analogous to the subspace constructed by M-LOBPCG, with the regular Ritz vectors replaced with the polynomial filtered Ritz vectors.

We observed that including the polynomial filtered Ritz vectors into the projection subspace could accelerate the convergence achieved by M-LOBPCG. If the preconditioned gradient cannot provide a good descent direction of the Rayleigh quotient for M-LOBPCG, the projected subspace involving polynomial filtered Ritz vectors tends to contain more accurate eigenvector approximations than the projected subspace for M-LOBPCG. Notice that if the degree of polynomial filter is set to be zero (no polynomial filter applied), our method is exactly the M-LOBPCG method. Also, we observed that with a poor preconditioner could significantly delay the convergence or even be counterproductive for the convergence of M-LOBPCG. However, our method could still converge robustly with such a poor preconditioner, thanks to the contribution by the polynomial filter. Compared with Chebyshev M-Davidson, including the preconditioned gradient into the projected subspace lower the demand for the quality of the polynomial filter, which could save a large number of matrix vector multiplications and is especially beneficial for LREP and CBSEP with dense matrices. In other words, our method could be a remedy if the polynomial filter is not effective for Chebyshev M-Davidson, or the preconditioner is not effective for M-LOBPCG. Also, we observed that if both the preconditioner and the polynomial filter are not effective, simultaneous utilization of them could be effective, which leads to stable convergence that might not be achieved by either of the two existing algorithms.

5.2.3 Chebyshev Polynomial Filter

We adopt the same shifted and scaled Chebyshev polynomial filter used in Chebyshev M-Davidson. Notice that the efficiency of the Chebyshev polynomial filter mainly depends on its

parameters, which are specified before Chebyshev M-LOBPCG starts. The polynomial degree d is a user-specified and fixed parameter. Estimates for a_0, a and b are needed, where $[a_0, a]$ contains the desired eigenvalues, whereas $[a, b]$ contains the undesired eigenvalues. The lower bound a of the undesired eigenvalue interval can be estimated and refined during an iterative process with minimal or no extra computational cost. For the upper bound b, it's recommended to use a much tighter bound by constructing an estimator using the Lanczos biorthogonalization procedure [92] for LREP. In our method, we use the M-Lanczos procedure applied to the matrix KM to obtain the initial values of a_0, a and b. During the iterations, we adaptively update a_0, a and b with the same strategy in [92]. Let ρ_{max}, ρ_{min}, ρ_{med} be the largest, the smallest and the median Ritz value at each step, respectively. Then, we may let $a \leftarrow \rho_{med}$. We may also set $b \leftarrow \rho_{max}$ or $a_0 \leftarrow \rho_{min}$, if $\rho_{max} > b$ or $\rho_{min} < a_0$. Since our target is the lowest k eigenvalues, without in-depth knowledge of the spectrum, it is not easy to select an accurate value of a to split the wanted and unwanted eigenvalues; fortunately, our choice works well in practice.

5.2.4 Choice of Search Direction

The choice of the search direction block $P^{(i)}$ deserves a careful discussion. Note that in our Chebyshev M-LOBPCG method, $P^{(i)}$ is constructed through the linear least squares, such that

$$P^{(i)} \subseteq \text{span}\{X_F^{(i)}, X_F^{(i-1)}\}. \tag{5.13}$$

For the simplicity of notation, let $F_d^{(i)}$ denote the polynomial filter adopted at the $i$the iteration. Then, after the Rayleigh-Ritz procedure at the ith step, we have three eigenvector approximations in hand, namely, $X^{(i)}$, $X_F^{(i)} = F_d^{(i)} X^{(i)}$ and $X_F^{(i-1)} = F_d^{(i-1)} X^{(i-1)}$. We could also involve $X^{(i)}$ into the search subspace to make a four-term recurrence, such that

$$\mathcal{S}^{(i)} = \text{span}\{X_F^{(i)}, W^{(i)}, X_F^{(i-1)}, X^{(i)}\}. \tag{5.14}$$

However, we noticed experimentally that this choice does not accelerate the convergence. On the other hand, the polynomial filter F_d will be updated after the Rayleigh-Ritz procedure at each step, thus, $F_d^{(i)} X_F^{(i-1)} = F_d^{(i)} F_d^{(i-1)} X^{(i-1)}$ tends to be a better eigenvector approximation than $X_F^{(i-1)}$,

and could be added into $P^{(i)}$. In this way, the search direction block $P^{(i)}$ can also be constructed as

$$P^{(i)} = X_F^{(i)} - F_d^{(i)} X_F^{(i-1)} C_X, \tag{5.15}$$

where C_X minimizes the 2-norm of each column of $X_F^{(i)} - F_d^{(i)} X_F^{(i-1)} C_X$, such that

$$P^{(i)} \subseteq \text{span}\{X_F^{(i)}, F_d^{(i)} X_F^{(i-1)}\} = \text{span}\{F_d^{(i)} X^{(i)}, F_d^{(i)} F_d^{(i-1)} X^{(i-1)}\}. \tag{5.16}$$

Also, $X^{(i)}$ is extracted by the Rayleigh-Ritz projection from the search subspace

$$\mathcal{S}^{(i-1)} = \text{span}\{X_F^{(i-1)}, W^{(i-1)}, P^{(i-1)}\}. \tag{5.17}$$

Then, we have $X^{(1)} \subseteq \text{span}\{F_d^{(0)} X^{(0)}, W^{(0)}\}$ and

$$X^{(2)} \subseteq \text{span}\{F_d^{(1)} F_d^{(0)} X^{(0)}, F_d^{(1)} W^{(0)}, W^{(1)}\}. \tag{5.18}$$

By induction, at the ith iteration, we have

$$X^{(i)} \subseteq \text{span}\left\{\prod_{j=0}^{i-1} F_d^{(j)} X^{(0)}, \prod_{j=1}^{i-1} F_d^{(j)} W^{(0)}, \prod_{j=2}^{i-1} F_d^{(j)} W^{(1)}, \cdots, F_d^{(i-1)} W^{(i-2)}, W^{(i-1)}\right\}. \tag{5.19}$$

This gives the same expression of the projection subspace as the ideal version of our proposed method, which exhibits the global quasi-optimality property. However, we observed numerically that such a choice of search directions could accelerate the rate of convergence but does not gain in runtime, because it involves one more application of polynomial filtering than the construction of the search direction block $P^{(i)}$ in Chebyshev M-LOBPCG at each step. We will further explain the difference between the ideal version and the practical version of our method in Section 5.3.

5.2.5 M-orthonormalization

In order to construct an M-orthonormal basis of the search subspace $\mathcal{S}^{(i)}$, we require the columns of $X_F^{(i)}$ to be M-orthonormal first. This can be done by first performing a Cholesky factorization of $X_F^{(i)T} M X_F^{(i)} = LL^T$, and then $X_F^{(i)} \leftarrow X_F^{(i)} L^{-T}$. When the columns of $X_F^{(i)}$ are close to be linearly dependent, the Cholesky factorization of $X_F^{(i)T} M X_F^{(i)}$ may break down numerically.

In this case, a more stable algorithm based on a truncated SVD of $X_F^{(i)T} M X_F^{(i)}$ may be used [39,56]. Also, a numerically stable Householder orthogonalization with a non-standard inner product is provided in [82].

After that, we need to M-orthogonalize the preconditioned residual block $W^{(i)} = T^{-1}R^{(i)}$ against $X_F^{(i)}$, which can be achieved by performing

$$W^{(i)} = W^{(i)} - X_F^{(i)}(X_F^{(i)T} M W^{(i)}). \tag{5.20}$$

Then, $W^{(i)}$ is M-orthonormalized using the Cholesky factorization based scheme described above. Finally, $P^{(i)}$ is made M-orthogonal against $Z^{(i)} = [X_F^{(i)}, W^{(i)}]$ by $P^{(i)} = P^{(i)} - Z^{(i)}(Z^{(i)T} M P^{(i)})$, and is then M-orthonormalized. Note that in our implementation, the above M-orthonormalization procedure is preformed twice at each iteration to keep the columns of the search subspace $\mathcal{S}^{(i)}$ being M-orthonormal to near machine precision, which is necessary for the algorithm to converge.

5.2.6 Implementation with Limited Memory

An implementation with efficient storage is necessary for eigenvalue algorithms, especially for large problems. In most implementations of eigenvalue algorithms, the block size is chosen at least as large as the number of desired eigenvalues. It is well known that block methods become less efficient when the block size is more than several dozens. A very large block size can lead to overwhelming memory demand. Even if the block size is not very large, setting the block size equal to the number of desired eigenvalues is still a challenge when the matrix is large. To overcome this challenge with limited memory, we employ a *hard deflation* technique such that the block size can be independent of and smaller than the number of desired eigenvalues. Hard deflation can be performed since the eigenvectors of KM are M-orthonormal. Suppose $X_c^{(i)}$ contains the converged eigenvectors in the ith iteration, and $S^{(i)}$ contains basis vectors of the original search subspace, the hard deflation can be done by

$$S^{(i)} = S^{(i)} - X_c^{(i)}(X_c^{(i)T} M S^{(i)}). \tag{5.21}$$

When some Ritz vectors have converged, new vectors are added into the eigenvector approximation block $X_F^{(i)}$ and the associated preconditioned residuals are added into $W^{(i)}$, respectively. Thus, the

sizes of $X_F^{(i)}$ and $W^{(i)}$ are always equal to the block size at the beginning and the size of the search directions $P^{(i)}$ varies during the iterations, since the converged Ritz vectors are excluded from it. The added vectors could be random vectors, but there is a better alternative. If the number of newly converged Ritz vectors is less than the block size, we could apply the polynomial filter on the unconverged columns of $X_F^{(i)}$ and then add them into $X_F^{(i)}$. In this way, no matter how many eigenvalues we desire, the block size could be flexible to choose (e.g.,1-20).

Also, for the sake of efficiency, we employ a *moving window* technique. We use one matrix V to store both the converged eigenvectors and the current subspace of Rayleigh-Ritz projection $\mathcal{S}^{(i)}$, namely, $V = [X_c^{(i)}, S^{(i)}]$. With more Ritz vectors converge and get locked, the size of $X_c^{(i)}$ becomes larger whereas $S^{(i)}$ keeps the three-block structure (5.9). This implementation has two advantages: (i) it avoids the memory copy once some eigenvectors converged, since the converged eigenvectors are already stored in the leading columns of V; (ii) orthogonalization of the blocks $W^{(i)}$ and $P^{(i)}$ to the converged eigenvectors $X_c^{(i)}$ and $X_F^{(i)}$ can be performed on one matrix. Notice that we may also store $MX_c^{(i)}$and $MS^{(i)}$ into one matrix, which optional, but it could save a few matrix vector multiplications when performing the M-orthogonalization.

A detailed description of our implementation is given in Algorithm 6.

5.3 Asymptotic global quasi-optimality of Chebyshev M-LOBPCG

In this section, we construct an ideal m-step Chebyshev M-LOBPCG method for computing the lowest eigenpair (λ_1, v_1) of the matrix pencil (MKM, M), and establish an *asymptotic global quasi-optimality* of this method. Our goal is to show that as the algorithm approaches convergence, its performance becomes increasingly similar to that of a corresponding globally optimal method. Such a property is analogous to the global optimality of the conjugate gradient method for the iterative solution of a symmetric and positive definite linear system.

5.3.1 An ideal m-step Chebyshev M-LOBPCG method

The framework of the ideal method is summarized in Algorithm 7. It uses the polynomial filter and preconditioner simultaneously to compute the lowest eigenpair of the matrix pencil (MKM, M). The motivation in exploring the optimality of this algorithm arises from [102], where the LOBPCG-like methods without using polynomial filters have been demonstrated to have the

Algorithm 6 Chebyshev M-LOBPCG

Input: SPD matrices $K, M \in \mathbb{R}^{n\times n}$, initial eigenvector approximations $X_0 \in \mathbb{R}^{n\times k}$, preconditioner $T^{-1} \in \mathbb{R}^{n\times n}$, number of desired eigenpairs $n_w > 0$, number of converged eigenpairs $n_c = 0$, maximum number of cycles $maxIter$, $X = M_X \in \mathbb{R}^{n\times(n_w+k)}$, $W = P = M_W = M_P \in \mathbb{R}^{n\times k}$.

Output: Smallest n_w eigenpairs $\{(\lambda_i, q_i)\}_{i=1}^{n_w}$ of the product matrix KM.

$X_0 \leftarrow F_d(KM)X_0$, $X(:, 1:k) \leftarrow M\text{-orth}\{X_0\}$, $M_X(:, 1:k) \leftarrow MX(:, 1:k)$;
$H \leftarrow M_X(:, 1:k)^T K M_X(:, 1:k)$;
Compute eigenpairs of H: $HY = Y\Lambda$;
$X(:, 1:k) \leftarrow X(:, 1:k)Y$; $M_X(:, 1:k) \leftarrow M_X(:, 1:k)Y$;
while $(j < maxIter)$ **do**
  $W \leftarrow KM_X(:, n_c + 1 : n_c + k) - X(:, n_c + 1 : n_c + k)\Lambda$;
  Test for convergence. Store converge Ritz pairs into λ_i and q_i. e_c is the number of newly converged Ritz pairs. $n_c = n_c + e_c$;
  if $(e_c > 0)$ **then**
    Generate new vectors $X_a \in \mathbb{R}^{n\times e_c}$;
    M-orthogonalize X_a against $X(:, 1 : n_c + k - e_c)$, $X_a \leftarrow M\text{-orth}\{X_a\}$;
    $X(:, n_c + k - e_c + 1 : n_c + k) \leftarrow X_a, M_X(:, n_c + k - e_c + 1 : n_c + k) \leftarrow MX_a$;
    Compute the eigenresiduals W_a, $W(:, 1 : e_c) \leftarrow W_a$;
  end if
  Apply the preconditioner $W \leftarrow T^{-1}W$. M-orthogonalize W against $X(:, 1 : n_c + k)$, $W \leftarrow M\text{-orth}\{W\}$, $M_W \leftarrow MW$;
  M-orthogonalize P to $[X(:, 1 : n_c + k), W]$, $P \leftarrow M\text{-orth}\{P\}$, $M_P \leftarrow MP$;
  $H \leftarrow [M_X(:, n_c + 1 : n_c + k), M_W, M_P]^T K [M_X(:, n_c + 1 : n_c + k), M_W, M_P]$;
  Compute eigenpairs of H: $HY = Y\Lambda$;
  $W \leftarrow [X(:, n_c + 1 : n_c + k), W, P]Y(:, 1 : k)$;
  Update a_0, a, b of the polynomial filter F_d, $W \leftarrow F_d(KM)W$;
  $P = W - X(:, n_c + 1 : n_c + k)X(:, n_c + 1 : n_c + k)^\dagger W$.
  M-orthogonalize W against $X(:, 1 : n_c)$, $W \leftarrow M\text{-orth}\{W\}$.
  $X(:, n_c + 1 : n_c + k) \leftarrow W$, $M_X(:, n_c + 1 : n_c + k) \leftarrow MW$;
end while

global quasi-optimality property. We will show that Algorithm 7 also exhibits such a favorable behavior when it approaches convergence.

Notice that we only provide the first cycle of the m-step Chebyshev M-LOBPCG method in Algorithm 7 for a fixed m. To obtain the final converged eigenvector approximation with this method, it requires a restart every m steps. Also, during this cycle, we assume that the polynomial filter F_d is fixed. Thus, in order to obtain the eigenvector approximation x_m at the end of the first cycle, the polynomial filter is applied $m(m+1)/2$ times, whereas it is applied only m times in Algorithm 6. The extensive use of polynomial filter involves a large number of matrix-vector multiplications, which makes Algorithm 7 too expensive to use in practice. Therefore, Algorithm 7 is the ideal version of Algorithm 6 with block size = 1 for computing the lowest eigenpair of the matrix pencil (MKM, M). More importantly, if the polynomial filter $F_d^{(i)}$ in Algorithm 6 is fixed and the block of search direction $P^{(i)}$ is generated satisfying (5.19), then the projection subspaces at the $m-1$st step in Algorithm 6 with block size = 1 and Algorithm 7 have the identical expression. Also, the global quasi-optimality of this ideal m-step algorithm is maintained only during the first m steps for a predetermined m.

Algorithm 7 ideal Chebyshev M-LOPCG for computing (λ_1, v_1) of (MKM, M)

1: Choose an SPD preconditioner $T^{-1} \approx (MKM - \sigma M)^{-1}$(where $\sigma < \lambda_1$), total number of steps $m > 0$.
2: Choose a polynomial filter F_d and vector x_0 with $\|x_0\|_M = 1$.
3: Compute $F_d^m x_0$ and scale the vector so that $\|F_d^m x_0\|_M = 1$, set $\rho_0 = \frac{F_d^m x_0^T MKMF_d^m x_0}{F_d^m x_0^T MF_d^m x_0}$, and $r_0 = (MKM - \rho_0 M)F_d^m x_0$
4: **for** $k = 0, \cdots, m-1$ **do**
5: Set $w_k = T^{-1} r_k$ and $p_k = F_d^{m-k-1} w_k$
6: **for** $j = 0, \cdots, k-1$ **do**
7: $p_k = p_k - \frac{p_k^T (MKM - \rho_k M) p_j}{p_j^T (MKM - \rho_k M) p_j} p_j$
8: **end for**
9: $p_k = \frac{p_k}{\|p_k\|_M}$
10: Form $S_k = [x_k, p_k]$, perform the Rayleigh-Ritz projection and solve $S_k^T MKMS_k u = \rho S_k^T MS_k u$ for the lowest primitive Ritz pair (ρ, u)
11: $x_{k+1} = S_k u$, $x_{k+1} = \frac{x_{k+1}}{\|x_{k+1}\|_M}$, $\rho_{k+1} = \frac{x_{k+1}^T MKMx_{k+1}}{x_{k+1}^T Mx_{k+1}}$, and $r_{k+1} = (MKM - \rho_{k+1} M)x_{k+1}$
12: **end for**

5.3.2 Global quasi-optimality

We first give the definition of the global quasi-optimality.

Definition 5.3.1. *Consider an iterative method for computing the lowest eigenpair (λ_1, v_1) of a symmetric and positive definite matrix pencil (MKM, M). Let x_0 be the initial vector, x_k be the approximation obtained at step k, and $\theta_k = \angle(x_k, v_1)_M = \cos^{-1} \frac{(v_1, x_k)_M}{\|v_1\|_M \|x_k\|_M}$ be the error angle of x_k. Let $U_1, U_2, \cdots$ be a sequence of subspaces of increasing dimension, such that for each $k \geq 1$, $x_k \in U_k$, and $U_i \subset U_j$ for $1 \leq i < j$. Let $y_k^* \in U_k$ be the global minimizer of the Rayleigh quotient $\rho(x) = \frac{x^T MKMx}{x^T Mx}$ in U_k. Then, for a given step k, the iterate x_k achieves the global quasi-optimality if*

$$\lim_{\theta_0 \to 0} \frac{\rho(x_k) - \rho(y_k^*)}{\rho(x_k) - \lambda_1} = 0. \tag{5.22}$$

Note that an iterative method with global quasi-optimality is nearly as good as the globally optimal method when approaching convergence. The difference between $\rho(x_k)$ computed by this method and $\rho(y_k^*)$ of global minimizer y_k^* in the same subspace is negligible compared to the error between $\rho(x_k)$ and the target eigenvalue λ_1, if the initial error angle θ_0 of x_0 is sufficiently small. Thus, the global quasi-optimality of an iterative eigenvalue algorithm is a strongly favorable property.

5.3.2.1 Linear convergence assumption

To show the global quasi-optimality of Algorithm 7, we first make an assumption of its *precisely* linear convergence.

Assumption 5.3.1. *Assume that Algorithm 7 starting with initial $\rho(x_0) \in (\lambda_1, \lambda_2)$ converges precisely linearly (not superlinearly or faster) to λ_1; in other words, there exists constants $0 < \underline{\xi} < \overline{\xi} < 1$, independent of the progress of Algorithm 7, such that*

$$\underline{\xi}^{k-j}(\rho_j - \lambda_1) \leq \rho_k - \lambda_1 \leq \overline{\xi}^{k-j}(\rho_j - \lambda_1), \quad for \ \ all \ \ 0 \leq j < k. \tag{5.23}$$

Also, we can rewrite Assumption 5.3.1 in terms of error angles. Consider two iterates $x_j = v_1 \cos\theta_j + f_j \sin\theta_j$ and $x_k = v_1 \cos\theta_k + f_k \sin\theta_k$, $0 \leq j < k$, where f_j, $f_k \perp Mv_1$, $\|f_j\|_M = \|f_k\|_M = 1$, and $\|x_j\|_M = \|x_k\|_M = 1$. Due to (2.3), we have (5.23) equivalent to $\underline{\xi}^{k-j} \sin^2\theta_j(\rho(f_j) - \lambda_1) \leq \sin^2\theta_k(\rho(f_k) - \lambda_1) \leq \overline{\xi}^{k-j} \sin^2\theta_j(\rho(f_j) - \lambda_1)$, i.e,

$$\sqrt{\underline{\xi}}^{k-j} \sqrt{\frac{\rho(f_j) - \lambda_1}{\rho(f_k) - \lambda_1}} \sin\theta_j \leq \sin\theta_k \leq \sqrt{\overline{\xi}}^{k-j} \sqrt{\frac{\rho(f_j) - \lambda_1}{\rho(f_k) - \lambda_1}} \sin\theta_j \tag{5.24}$$

Therefore, we can conclude that $\mathcal{O}(\sin\theta_0) = \mathcal{O}(\sin\theta_l)(1 \leq l \leq k)$ for a sufficiently small θ_0 and a fixed k.

Next, we make an assumption on the effect of the polynomial filter as follows.

Assumption 5.3.2. *Assume that F is a polynomial filter in Algorithm 7. Let y have the decomposition $y = v_1\cos\theta_y + f_y\sin\theta_y \in U$, where $f_y \perp Mv_1$, and $\|f_y\|_M = 1$ and Fy have the decomposition $Fy = v_1\cos\theta_{Fy} + f_{Fy}\sin\theta_{Fy} \in FU$, where $f_{Fy} \perp Mv_1$, and $\|f_{Fy}\|_M = 1$ then we have*

$$\rho(y) \geq \rho(Fy), \tag{5.25}$$

$$\sin\theta_{Fy} = \mathcal{O}(\sin\theta_y). \tag{5.26}$$

Assumption 5.3.2 assumes that application of the polynomial filter F to the vector y is guaranteed to decrease its Rayleigh quotient, but such an improvement is moderate. In other words, F makes Fy closer to the target eigenvector v_1 than y, meanwhile, the error angle of Fy cannot be dramatically smaller than that of y. This assumption is largely consistent with our numerical experience with many test matrices.

5.3.2.2 Relevant spaces

To study the global quasi-optimality of Algorithm 7, we define

$$W_k = \text{span}\{p_0, p_1, \cdots, p_{k-1}\}, \tag{5.27}$$

and

$$U_k = \text{span}\{F^m x_0\} + W_k. \tag{5.28}$$

Proposition 5.3.1. *For all $1 \leq k \leq m$ in Algorithm 7, $W_k = \text{span}\{F^{m-1}w_0,$ $F^{m-2}w_1, \cdots, F^{m-k}w_{k-1}\}$ and $x_k \in U_k$. Also, $p_k^T(MKM - \rho_k M)p_j = 0$ for $0 \leq j \leq k-1$..*

Proof. It is easy to check that $W_k = \text{span}\{F^{m-1}w_0, F^{m-2}w_1, \cdots, F^{m-k}w_{k-1}\}$ by the step 5 of Algorithm 7. The rest could be done by induction. We have $x_1 \in \text{span}\{F^m x_0, F^{m-1}w_0\} = \{F^m x_0, p_0\} = U_1$. For a given k, x_k is extracted from the subspace $S_k = \text{span}\{x_{k-1}, p_{k-1}\}$ by Rayleigh-Ritz, where p_{k-1} is a linear combination of $F^{m-k-2}w_{k-1}$ and $\{p_0, p_1, \cdots, p_{k-2}\}$. Then, we conclude that

$x_k \in U_k$. Also, when we construction the search directions $\{p_k\}$ at the step 7 of Algorithm 7, $p_k^T(MKM-\rho_k M)p_j = 0$ for $0 \le j \le k-1$ is achieved by the explicit orthogonalization of p_k against p_j's with respect to the scalar product involving $MKM - \rho_k M$. □

Remark. After m steps, we have

$$x_m \in U_m = \text{span}\{F^m x_0, F^{m-1}w_0, F^{m-2}w_1, \cdots, Fw_{m-2}, w_{m-1}\}, \tag{5.29}$$

which has the identical expression with (5.19), given that $F_d^{(j)}$ is fixed there.

5.3.2.3 Preliminary results

We present a few preliminary results [102, Assumption 4.4, Lemma 3-4] useful for the proof of the main theorems.

Assumption 5.3.3. *Assume that there is a constant $\delta > 0$, independent of $\theta_0 = \angle(x_0, v_1)_M$, such that $\angle(v_1, W_k) \ge \delta$ for all $k \ge 1$.*

Lemma 5.3.1. *Let $x \approx v_1$ in (2.1) , and p be a descent direction for $\rho(x)$ such that $(p, \nabla\rho(x)) < 0$. Assume that there exists a $\delta > 0$ independent of $\theta = \angle(x, v_1)_M$, such that $\rho(p) - \rho(x) \ge \delta$. Then, the optimal step size α^* minimizing $\rho(x + \alpha p)$ is the unique or the smaller positive root of $a(x,p)\alpha^2 + b(x,p)\alpha + c(x,p) = 0$, where*

$$\begin{aligned}
a(x,p) &= (p^TMKMp)(p^TMx) - (p^TMp)(p^TMKMx) \\
&= \|p\|_M^2 x^T(MKM - \rho(p)M)p \\
b(x,p) &= (p^TMKMp)(x^TMx) - (p^TMp)(x^TMKMx) \\
&= \|x\|_M^2 p^T(MKM - \rho(x)M)p \\
&= \|x\|_M^2\|p\|_M^2(\rho(p) - \rho(x)) \ge \|x\|_M^2\|p\|_M^2\delta > 0, \text{ and} \\
c(x,p) &= (p^TMKMx)(x^TMx) - (p^TMx)(p^TMKMx) \\
&= \|x\|_M^2 p^T(MKM - \rho(x)M)x < 0.
\end{aligned} \tag{5.30}$$

Lemma 5.3.2. *Let x be an approximation to v_1 with decomposition (2.1) , p be a descent direction for $\rho(x)$, and $\delta > 0$ be a constant independent of $\theta = \angle(x, v_1)_M$, such that $\rho(p) - \rho(x) \ge \delta > 0$, and*

α^* the optimal step size minimizing $\rho(x+\alpha p)$. Then for sufficiently small θ, with a, b and c defined in (5.30),

$$\frac{-\frac{c^2}{2b}-\frac{ac^3}{3b^3}+\mathcal{O}(c^4)}{\min\{\|x\|_M^2,\|x+\alpha^* p\|_M^2\}}\leq \rho(x+\alpha^* p)-\rho(x)\leq \frac{-\frac{c^2}{2b}-\frac{ac^3}{3b^3}+\mathcal{O}(c^4)}{\max\{\|x\|_M^2,\|x+\alpha^* p\|_M^2\}}, \tag{5.31}$$

and hence $\rho(x+\alpha^* p)-\rho(x)=-\mathcal{O}(\sin^2\theta)\mathcal{O}(\cos^2\angle(p,\nabla\rho(x)))$.

Next, we prove a lemma that provides us the decomposition of x_k in Algorithm 7. The proof is in the Appendix.

Lemma 5.3.3. *For $1\leq k\leq m$ in Algorithm 7. Consider two iterates $x_0=v_1\cos\theta_0+ f_0\sin\theta_0\in U_0$, and $x_k=v_1\cos\theta_k+f_k\sin\theta_k\in U_k$, where f_0, $f_k\perp Mv_1$, $\|f_0\|_M=\|f_k\|_M=1$, $\|x_0\|_M=\|x_k\|_M=1$ with sufficiently small θ_0, θ_k such that $\lambda_1<\rho(x_k)\leq\rho(x_0)<\lambda_2$. Consider a decomposition*

$$x_k=\beta_{0k}F^m x_0+\gamma_{0k}g_{0k}, \tag{5.32}$$

with $\beta_{0k},\gamma_{0k}>0$, $\|F^m x_0\|_M=\|g_{0k}\|_M=1$, $g_{0k}\in\operatorname{span}\{p_0,p_1,\cdots,p_{k-1}\}$. Let $\mu_{0k}=g_{0k}^T F^m x_0$, then we have for some scalar $q\geq 1$,

$$\gamma_{0k}=\mathcal{O}(\sin^q\theta_0),\ \beta_{0k}=1-\mathcal{O}(\sin^q\theta_0). \tag{5.33}$$

5.3.2.4 Main theorems

We are ready to prove the global quasi-optimality of Algorithm 7. First, we show that the quality of the iterate of Algorithm 7 at step k for approximating the corresponding global minimizer can be extended to step $k+1$ with a possible very small deterioration on the order of $\mathcal{O}(\sin\theta_{k+1})\mathcal{O}(\sin^2\theta_0)$. This theorem plays a foundational role in our effort to establish the global quasi-optimality of Algorithm 7. The proof is long and technical, therefore is given in the Appendix.

Theorem 5.3.1. *Let $\{x_k\}$ be the iterates of Algorithm 7 and $\{p_k\}$ be the M-normalized search directions. For a given $0\leq k\leq m-1$, consider all vectors of the form $z=y+\alpha p_k\in U_{k+1}$, where $y=\beta_y F^m x_0+\gamma_y g_y\in U_k$ with $\|y\|_M=\|g_y\|_M=\|F^m x_0\|_M=1$, and $g_y\in W_k$. Let y_k^* and y_{k+1}^* be the global minimizer of $\rho(\cdot)$ in U_k and U_{k+1}, respectively. Then*

$$\rho(x_{k+1})-\rho(y_{k+1}^*)\leq\rho(x_k)-\rho(y_k^*)+\mathcal{O}(\sin\theta_{k+1})\mathcal{O}(\sin^2\theta_0). \tag{5.34}$$

Now, we can present the major theorem of this section on the global quasi-optimality of all iterates generated by Algorithm 7.

Theorem 5.3.2. *Let $\{x_k\}$ be the iterate of Algorithm 7 and $\{p_k\}$ be the M-normalized search directions. Suppose that Assumptions 5.3.1, 5.3.2 and 5.3.3 hold. In Theorem 5.3.1, let y_k^* and y_{k+1}^* be the global minimizer in U_k and U_{k+1}, respectively, and C_{k+1} be the constant for which $\rho(x_{k+1}) - \rho(y_{k+1}^*) \leq \rho(x_k) - \rho(y_k^*) + C_{k+1} \sin\theta_{k+1} \sin^2\theta_0$. Then, for any given $2 \leq k \leq m-1$, $\rho(x_k) - \rho(y_k^*) \leq \sum_{j=2}^{k} C_j \sin\theta_j \sin^2\theta_0$, and therefore,*

$$\lim_{\theta_0 \to 0} \frac{\rho(x_k) - \rho(y_k^*)}{\rho(x_k) - \lambda_1} \leq \lim_{\theta_0 \to 0} \frac{\sum_{j=2}^{k} C_j \sin\theta_j \sin^2\theta_0}{\underline{\xi}^k (\rho(f_0) - \lambda_1) \sin^2\theta_0} \leq \lim_{\theta_0 \to 0} \frac{\sum_{j=2}^{k} C_j \sin\theta_j}{\underline{\xi}^k (\lambda_2 - \lambda_1)} = 0. \tag{5.35}$$

Proof. The proof is based on mathematical induction.

Since $x_1 \in \text{span}\{F_m x_0, p_0\} = U_1$ is obtained from the Rayleigh-Ritz projection, it is the global minimizer in U_1. Let y_2^* be the global minimizer of $\rho(\cdot)$ in U_2. It follows from Theorem 5.3.1 that

$$\rho(x_2) - \rho(y_2^*) \leq \rho(x_1) - \rho(y_1^*) + \mathcal{O}(\sin\theta_2)\mathcal{O}(\sin^2\theta_0) = \mathcal{O}(\sin\theta_2)\mathcal{O}(\sin^2\theta_0),$$

where $x_1 = y_1^*$ is the global minimizer in U_1. Then, the base case is established, i.e, x_2 is a global quasi-minimizer in U_2 such that $\lim_{\theta_0 \to 0} \frac{\rho(x_2) - \rho(y_2^*)}{\rho(x_2) - \lambda_1} = 0$. Suppose that $\rho(x_{k-1}) - \rho(y_{k-1}^*) \leq \sum_{j=2}^{k-1} C_j \sin\theta_j \sin^2\theta_0$ holds. Then, by Theorem 5.3.1 we have $\rho(x_k) - \rho(y_k^*) \leq \rho(x_{k-1}) - \rho(y_{k-1}^*) + \mathcal{O}(\sin\theta_k)\mathcal{O}(\sin^2\theta_0) \leq \sum_{j=2}^{k} C_j \sin\theta_j \sin^2\theta_0$, as desired. We know $f_0 \perp Mv_1$, then $\rho(f_0) \geq \lambda_2$. Also, we have $\rho_0 - \lambda_1 = (\rho(f_0) - \lambda_1)\sin^2\theta_0$ and $\underline{\xi}^k(\rho_0 - \lambda_1) \leq \rho_k - \lambda_1$, thus the following inequalities hold, i.e.,

$$\lim_{\theta_0 \to 0} \frac{\rho(x_k) - \rho(y_k^*)}{\rho(x_k) - \lambda_1} \leq \lim_{\theta_0 \to 0} \frac{\sum_{j=2}^{k} C_j \sin\theta_j \sin^2\theta_0}{\underline{\xi}^k (\rho(f_0) - \lambda_1) \sin^2\theta_0} \leq \lim_{\theta_0 \to 0} \frac{\sum_{j=2}^{k} C_j \sin\theta_j}{\underline{\xi}^k (\lambda_2 - \lambda_1)} = 0.$$

□

5.4 Numerical experiments

In this section, we perform extensive numerical experiments to validate our algorithm and demonstrate its effectiveness compared with other algorithms. Since we focus on methods with

limited memory for large problems, we control the block size. We implement all the methods based on *hard deflation* and the *moving window* techniques discussed in section 5.2.6. All experiments seek $n_w = 150$ lowest eigenvalues with block size $k = 10$. To make fair comparisons, the maximum size of projection subspace for all methods is set to be $3k = 30$, which is the standard subspace size of LOBPCG. The maximum number of iterations is set to be 10000. All experiments use a stopping tolerance $tol = 10^{-7}$ for the relative eigenresidual of (1.25) or (1.32), such that

$$\frac{\|KMx_i - \rho_i x_i\|_2}{|\rho_i|\|x_i\|_2} \leq tol.$$

The Chebyshev polynomial filter degrees $polyN = 5, 10, 20$ have been considered. All methods (except LOBP4DCG) start with the same initial approximation, with random entries uniformly distributed on $[0, 1]$, generated by MATLAB's `rand` function with the fixed seed (`rng default`). Note that LOBP4DCG could be extended to solve the product eigenvalue problems in practice. To perform a complete comparison, we also solve the CBSEP with LOBP4DCG. Although LOB4DPCG does not work directly on the product eigenvalue problem for LREP, it is sufficient to check the convergence of n elements of the eigenvector approximation. Also, the LOBP4DCG method should start with the initial approximation with twice as many random entries as that for other algorithms, also constructed by the `randn` function. All our experiments were performed on a Windows 10 (64 bit) PC-Intel(R) Core(TM) i7-4700 CPU 2.40 GHz, 32 GB of DDR3 1600MHz RAM running MATLAB R2022b.

5.4.1 Test Problems

The test problems are given in Table 5.1. Test 1 to Test 3 come from linear response analysis for disodium (Na2, Na4), and silane (SiH4) compounds, respectively. Test 4 to Test 6 correspond to discretized Bethe–Salpeter Hamiltonians associated with boron nitride (BN). Also, we provide the ratio between the the width of the wanted spectrum against the width of the unwanted spectrum for each test problem as

$$\frac{\lambda_{n_w} - \lambda_{\min}}{\lambda_{\max} - \lambda_{n_w+1}}, \tag{5.36}$$

which affects the quality of the polynomial filter. Generally, for a fixed Chebyshev polynomial filter degree, a smaller ratio (5.36) makes the the polynomial filter less effective. In other words, given a test problem with a small ratio (5.36), the polynomial filter based algorithms require a high polynomial degree to be effective. As Table 5.1 shows, the problem Na4 is considerably more demanding for polynomial filters than other test problems.

Table 5.1: Properties of the test matrices

Problem	size n	ratio (5.36)
Test 1 (Na2)	1862	3.03e-02
Test 2 (Na4)	2834	2.05e-03
Test 3 (SiH4)	5660	2.77e-02
Test 4 (BN1)	4608	3.98e-02
Test 5 (BN2)	4608	2.00e-02
Test 6 (BN3)	8192	5.20e-02

5.4.2 An Illustration Example of Quasi-optimality for the Ideal Chebyshev M-LOBPCG(Algorithm7)

The global quasi-optimality as defined in (5.22) implies that as the initial approximation becomes closer to the eigenvector associated with the desired lowest eigenvalue, the relative difference between the Rayleigh quotient obtained at step i of Algorithm 7 and that obtained at the same step of the globally optimal algorithm tends to approach zero. Figure 5.1 demonstrates that Algorithm 7 achieves the global quasi-optimality on two test problems. For both problems, we choose step $m = 20$, with incomplete Cholesky decomposition preconditioner with drop tolerance $s = 0.01$ and the degree of polynomial filter $polyN = 5$. We plot the quasi-optimal ratio (5.22) against the number of iterations in Figure 5.1. It shows that as the initial approximation approaches the desired eigenvector, at any given step k, the relative difference between the Rayleigh quotients generated by Algorithm 7 and the globally optimal method becomes smaller. This suggests that, given a fixed step k, the convergence of the ideal Chebyshev M-LOBPCG becomes increasingly close to the globally optimal method as the ideal algorithm converges towards the desired eigenpair. Meanwhile, given an initial approximation x_0 with a fixed θ_0, as the step k increases, the ratio (5.22) increases to 1, showing that the globally optimal method becomes progressively more superior to the locally optimal Algorithm 7 as both methods proceed. This is also largely consistent with our extensive numerical experience with locally optimal eigenvalue algorithms.

Although the ideal Chebyshev M-LOBPCG has such a favorable property, we should point out that it is impractical to use this algorithm to solve the test problems, which is not only expensive in the polynomial filter applications count ($m(m-1)/2$), but also very expensive with the search direction orthogonalization cost (proportional to m^2). On the other hand, the global quasi-optimality can be kept only for m steps with a predetermined value of m, which makes it *not* realistic to select an appropriate m to gain in runtime compared with our Chebyshev M-LOBPCG.

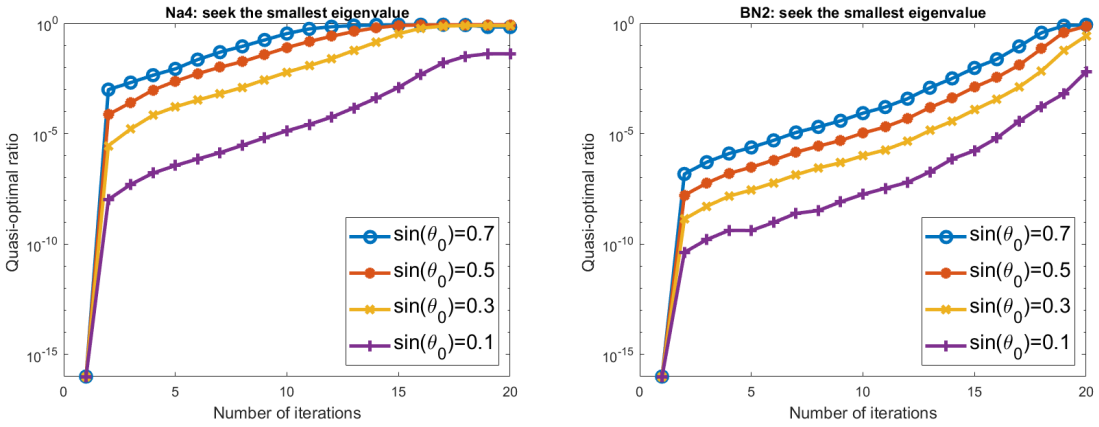
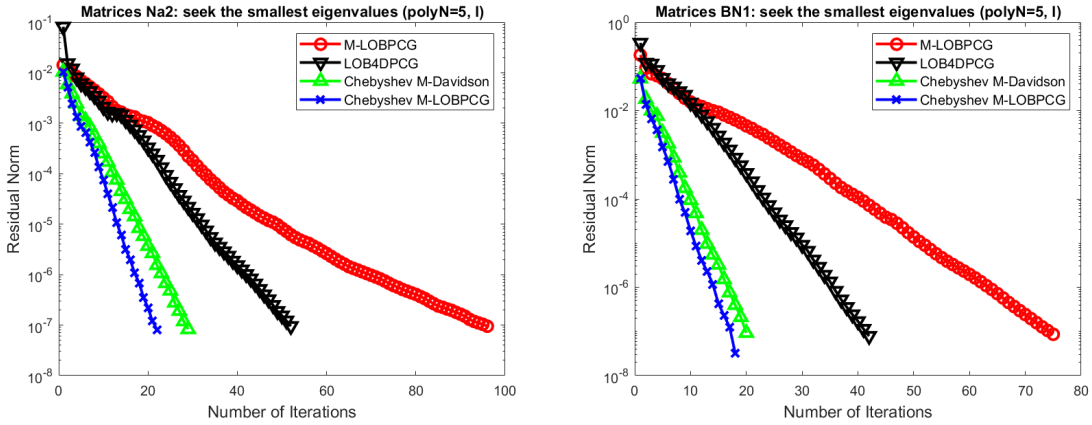


Figure 5.1: Confirming the quasi-optimality of Algorithm 7 showing that the ratio (5.22) converges to zero as the angle of the initial guess to the required eigenvector decreases.

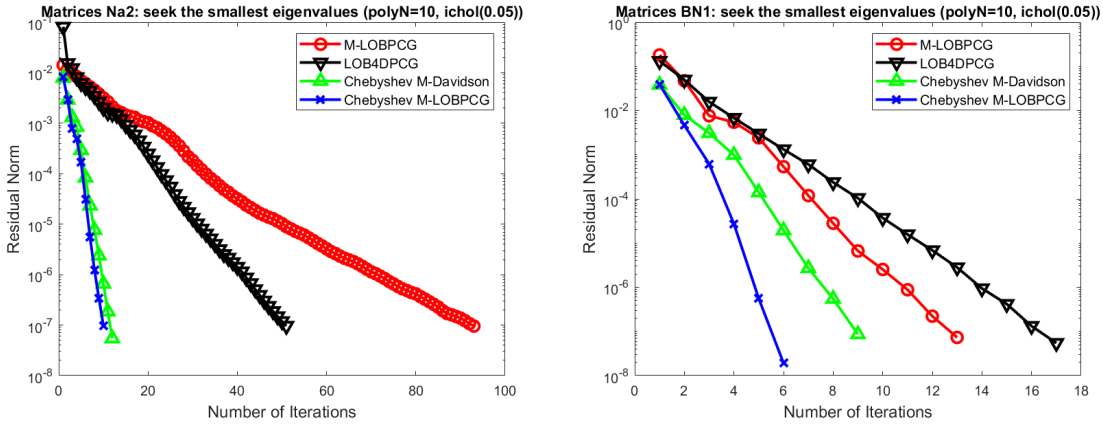
5.4.3 An Illustration Example of Convergence

Figure 5.2 and Figure 5.3 shows the convergence towards the smallest eigenvalue for problems Na2 and BN1. For both problems, Chebyshev M-LOBPCG exhibits fastest convergence. Also, with *polyN* increasing from 5 to 10, Chebyshev M-Davidson and Chebyshev M-LOBPCG achieve faster convergence for both problems. For these problems we chose, the Chebyshev polynomial filter seems more effective than the preconditioner we used; however, this might not be always true. For those problems that the preconditioner is more effective than the polynomial filter, our method could also take advantage of the preconditioner. For Na2, with the preconditioner applied to the gradient, it seems like that M-LOBPCG and LOBP4DCG could not achieve very fast convergence. For BN1, with the preconditioner applied to the gradient, M-LOBPCG and LOBP4DCG were able to converge more rapidly. This example also demonstrate the robustness of Chebyshev M-LOBPCG, which is less sensitive to the quality of the polynomial filter and the preconditioner applied to the gradient.



(a) Problem Na2 (b) Problem BN1

Figure 5.2: Comparison between Chebyshev M-LOBPCG and other methods for computing the smallest eigenvalue with polyN=5 and without preconditioner.



(a) Problem Na2 (b) Problem BN1

Figure 5.3: Comparison between Chebyshev M-LOBPCG and other methods for computing the smallest eigenvalue with polyN=10 and incomplete Cholesky decomposition preconditioner with drop tolerance $s = 0.05$.

5.4.4 Comparison with Chebyshev M-Davidson method

We compare our method against the Chebyshev M-Davidson method. Table 5.2, 5.3 and 5.4 summarize results based on the degree $polyN$ of the polynomial filters. In each table, "iter" counts the total number of iterations, "mvp" is the number of matrix-vector products, and "time" is the runtime in seconds. "NA" implies that we fail to obtain all the desired eigenvalues within 10000 iterations. An algorithm using the incomplete Cholesky factorization preconditioner with drop tolerance s is denoted by the name of the algorithm followed by (s), and an unpreconditioned method is denoted by its name followed by (I). For example, in Table 5.2, for the test problem Sih4, the first three numbers shows that given a polynomial filter of degree $polyN = 5$, the Chebyshev M-Davidson method required 8530 iterations and 1492603 matrix-vector products to find all the wanted eigenvalues in 3092.67 seconds. We observe that our method converges faster in terms of number of iterations and matrix-vector multiplications given the same $polyN$ even without a preconditioner. Adding a preconditioner of low to modest quality could further accelerate convergence. For the LREP problems Na2, Sih4 and the CBSEP problems BN1, BN3, which only require a relatively low $polyN$ for Chebyshev M-Davidson to be effective, our method matches or outperforms it in terms of the runtime. For the LREP problem Na4 and the CBSEP problem BN2, Chebyshev M-Davidson requires a relatively high $polyN$ to be effective, which is consistent with the ratio (5.36) we provide in Table 5.1, but our method still achieved fast convergence given a moderate $polyN$. In fact, the harder the problem is for Chebyshev M-Davidson, the more significant relative advantage our method has. In addition, the overall performance of our method is less sensitive to the degree of polynomial filters than the Chebyshev M-Davidson method.

Table 5.2: Performance comparison of Chebyshev M-Davidson and Chebyshev M-LOBPCG ($polyN = 5$).

Matrix	Chebyshev M-Davidson			Chebyshev M-LOBPCG(I)			Chebyshev M-LOBPCG(10^{-2})		
	mvp	iter	time	mvp	iter	time	mvp	iter	time
Na2	294541	1684	72.21	173840	784	51.40	169660	765	50.51
Na4	NA	NA	NA	1772360	8050	1061.10	1378100	6258	842.92
Sih4	1492603	8530	3092.67	234560	1060	553.56	146100	658	344.29
BN1	459415	2626	622.17	202440	914	311.22	110480	496	171.41
BN2	642099	3670	871.41	251060	1135	363.62	110690	497	169.10
BN3	459398	2626	2812.58	198920	898	1368.63	174500	787	1205.99

Table 5.3: Performance comparison of Chebyshev M-Davidson and Chebyshev M-LOBPCG ($polyN = 10$).

Matrix	Chebyshev M-Davidson			Chebyshev M-LOBPCG(I)			Chebyshev M-LOBPCG(10^{-2})		
	mvp	iter	time	mvp	iter	time	mvp	iter	time
Na2	131340	478	29.64	100130	304	29.04	85730	259	26.49
Na4	1837386	6682	972.06	1066530	3324	608.50	668450	2080	387.31
Sih4	617253	2245	1274.89	124750	381	297.78	99770	303	243.83
BN1	220714	803	297.27	108750	331	172.38	88610	268	142.51
BN2	357928	1302	481.42	172770	531	264.68	105250	320	168.72
BN3	198920	898	1370.97	122850	375	880.26	114190	348	824.83

Table 5.4: Performance comparison of Chebyshev M-Davidson and Chebyshev M-LOBPCG ($polyN = 20$).

Matrix	Chebyshev M-Davidson			Chebyshev M-LOBPCG(I)			Chebyshev M-LOBPCG(10^{-2})		
	mvp	iter	time	mvp	iter	time	mvp	iter	time
Na2	93004	196	21.03	78110	139	25.36	73950	131	22.95
Na4	793102	1670	435.53	499790	950	304.66	448870	852	273.63
Sih4	150482	317	317.47	99230	180	254.66	88350	159	227.95
BN1	136224	287	184.08	80630	144	139.56	76510	126	123.83
BN2	189892	400	256.39	124910	229	204.51	97230	176	161.12
BN3	113446	239	723.59	92670	167	720.73	87470	157	680.49

5.4.5 Comparison with LOBPCG-type method

We compare our method against M-LOBPCG method and LOBP4DCG. Table 5.5, 5.6 and 5.7 summarize results based on the preconditioner on the gradient. For our method, we show the results with a moderate degree $polyN = 10$. For example, in Table 5.6, for the test problem Sih4, the first three numbers shows that the LOBP4DCG method with an incomplete Cholesky factorization ($s = 10^{-2}$) as the preconditioner, required 1209 iterations and 96934 matrix-vector products to find all the wanted eigenvalues in 264.01 seconds. We observe that the M-LOBPCG method converge very slowly for Na4, Sih4 and BN2 without a preconditioner or with an incomplete Cholesky decomposition preconditioner with drop tolerance $s = 10^{-2}$. Also, it seems that LOB4DPCG is not efficient for computing a large number of eigenvalues. We should point out that the performance of M-LOBPCG tends to rely on the quality of the preconditioner significantly. A large drop tolerance e.g. 10^{-2} of the incomplete Cholesky factorization may delay the convergence for the test problems Sih4 and BN2, which does not occur to LOB4DPCG and our method.

From these results, we can see that our method converges faster than other LOBPCG-type methods in terms of number of iterations, but may involve more matrix-vector multiplications

occasionally for certain difficult problems such as Na4, when the spectral width ratio (5.36) is so small that polynomial filters are not very effective. However, in typical high-performance computing environments these matrix-vector multiplications can be evaluated more efficiently but incomplete factorizations may encounter performance bottlenecks, we assume that our method is likely more efficient. Tables 5.6 and 5.7 show that with a good preconditioner, the number of iterations for all methods decreases, but the additional cost of applying the preconditioner per iteration may lead to a significant increase in computational time. In this regard, our method could be a better choice, since it does not require a preconditioner of high quality that might be expensive to use. For the problem Na4, Tables 5.6 and 5.7 clearly show that if the polynomial filter is not quite efficient, using a preconditioner of higher quality can effectively compensate for the performance of Chebyshev M-LOBPCG.

Table 5.5: Performance comparison of LOBPCG-type methods and Chebyshev M-LOBPCG ($polyN = 10$).

Matrix	LOBP4DCG(I)			M-LOBPCG(I)			Chebyshev M-LOBPCG(I)		
	mvp	iter	time	mvp	iter	time	mvp	iter	time
Na2	185738	2319	99.31	546480	4969	155.03	100130	304	29.04
Na4	300456	3753	282.90	NA	NA	NA	1066530	3324	608.50
Sih4	194378	2427	508.02	519750	4726	1189.70	124750	381	297.78
BN1	213578	2667	623.99	474120	3952	695.72	108750	331	172.38
BN2	221338	2764	642.20	738960	6159	1076.73	172770	531	264.68
BN3	422058	5273	3694.90	481800	4016	3220.09	122850	375	880.26

Table 5.6: Performance comparison of LOBPCG-type methods and Chebyshev M-LOBPCG ($polyN = 10$).

Matrix	LOBP4DCG(10^{-2})			M-LOBPCG(10^{-2})			Chebyshev M-LOBPCG(10^{-2})		
	mvp	iter	time	mvp	iter	time	mvp	iter	time
Na2	185258	2313	100.90	318450	2896	93.07	85730	259	26.49
Na4	146776	1832	147.47	NA	NA	NA	668450	2080	387.31
Sih4	96934	1209	264.01	NA	NA	NA	99770	303	243.83
BN1	112938	1409	335.43	80160	669	120.19	88610	268	142.51
BN2	127978	1597	381.76	NA	NA	NA	105250	320	168.72
BN3	327338	4089	2841.46	211080	1760	1442.44	114190	348	824.83

Table 5.7: Performance comparison of LOBPCG-type methods and Chebyshev M-LOBPCG ($polyN = 10$).

Matrix	LOBP4DCG(10^{-5})			M-LOBPCG(10^{-5})			Chebyshev M-LOBPCG(10^{-5})		
	mvp	iter	time	mvp	iter	time	mvp	iter	time
Na2	88696	1106	138.17	47850	436	48.84	65890	197	34.18
Na4	77096	961	141.67	44220	403	55.25	130530	399	106.65
Sih4	87654	1093	692.30	52580	479	326.80	82750	250	300.66
BN1	115338	1439	677.25	73920	617	257.05	79970	241	183.99
BN2	85658	1068	501.17	53520	447	193.80	89890	272	223.81
BN3	322298	4026	5276.83	205560	1714	2461.78	119010	363	1078.71

5.5 More experiments for the symmetric eigenvalue problem

In this section, we extend our Chebyshev M-LOBPCG method (Algorithm 6) to the the symmetric eigenvalue problem such that

$$Ax = \lambda x \tag{5.37}$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric and $(\lambda,\ x)$ is the corresponding eigenpair. Note that in the BSE problem, we consider the product eigenvalue problem (1.22), where K and M are symmetric and positive definite. Assume that $M = I$ and $K = A + \sigma I$, where I is the identity matrix and σ is the shift parameter such that K is symmetric and positive definite. Then, the Chebyshev M-LOBPCG method (Chebyshev LOBPCG) can be simply employed for the symmetric eigenvalue problem (5.37).

For the rest of this section, we perform extensive experiments to compare our Chebyshev LOBPCG method against the LOBPCG [63] and Chebyshev Davidson methods [110]. The test matrices are available from the Suite Sparse Matrix Collection [65] except two Laplacian matrices generated artificially. The information of the test problems is given in Table 5.8, where we record the problem size n, the number of nonzero elements nnz and the application field. Also, we provide the ratio between the the width of the wanted spectrum against the width of the unwanted spectrum for each test problem (5.36), which affects the quality of the polynomial filter. To make fair comparisons, the maximum size of projection subspace for all methods are the same. All methods start with the same initial approximation, with random entries uniformly distributed on $[0, 1]$, generated by MATLAB's `rand` function with the fixed seed (`rng default`). All experiments use a stopping

tolerance $tol = 10^{-7}$ for the relative eigenresidual of (5.37), such that

$$\frac{\|Ax_i - \lambda_i x_i\|_2}{|\lambda_i| \|x_i\|_2} \leq tol.$$

When the test matrix A is symmetric indefinite, we perform 30 Lanczos steps to find a rough approximate of the leftmost eigenvalue, say $\widehat{\lambda_{left}}$. Then, the shift parameter σ is set to be $1.5|\widehat{\lambda_{left}}|$ such that $A + \sigma I$ is positive definite. Other parameter settings are the same as those in Section 5.4 for the BSE problem.

We summarize the results in Table 5.9 and 5.10 based on the number of wanted eigenvalues n_w. The degree $polyN$ of the polynomial filters is set to be 20, when we apply the polynomial filters in an algorithm. We still use the incomplete Cholesky factorization as the preconditioner applied on the gradient and the drop tolerance is set to be 10^{-2}. In Table 5.9, we compute 5 wanted eigenvalues and the block size k is set to be 5. We compute 100 wanted eigenvalues with $k = 20$ in Table 5.10. For each test problem, the method with the least run-time is highlighted in bold. From these results, we can see that our method could achieve the best or second best performance in terms of the run-time. Also, the fastest convergence in terms of the number of iterations is achieved with our method. We observe that our method could still perform well, even though the quality of the polynomial filter and the preconditioner applied on the gradient is poor at the same time. Overall, our Chebyshev LOBPCG method is still competitive for the symmetric eigenvalue problem given a relatively low degree of polynomial filter and a preconditioner of low-to-modest quality.

Table 5.8: Properties of the test matrices

Matrix	n	nnz	ratio($n_w = 5$)	ratio ($n_w = 100$)	Source
Tre20k	20000	554466	4.28e-05	2.40e-03	Combinatorial Problem
finan512	74752	596992	3.00e-03	9.31e-03	Economic Problem
hb1138bus	1138	4054	5.95e-06	7.50e-05	Power Network Problem
cfd1	70656	1825580	2.87e-05	3.25e-04	Computational Fluid Dynamics
Muu	7102	170134	1.70e-08	1.14e-02	Structural Problem
wathen100	30401	471601	1.82e-03	8.24e-03	Random 2D/3D Problem
obstclae	40000	197608	1.51e-04	3.21e-03	Optimization Problem
shallowwater1	81920	327680	8.27e-04	5.33e-03	Computational Fluid Dynamics
minsurfo	40806	203622	1.54e-05	6.48e-04	Optimization Problem
Ga41As41H72	268096	18488476	3.71e-05	4.66e-04	Theoretical/Quantum Chemistry
GaAsH6	61349	3381809	4.39e-04	1.67e-03	Theoretical/Quantum Chemistry
Ga3As3H12	61349	5970947	2.21e-04	1.53e-03	Theoretical/Quantum Chemistry
Si87H76	240369	10661631	1.65e-03	1.20e-02	Theoretical/Quantum Chemistry
Ge99H100	112985	8451395	1.75e-03	1.44e-02	Theoretical/Quantum Chemistry
SiO2	155331	11283503	4.73e-03	1.33e-02	Theoretical/Quantum Chemistry
CO	221119	7666057	1.48e-02	2.38e-02	Theoretical/Quantum Chemistry
Si41Ge41H72	185639	15011265	1.20e-03	1.11e-02	Theoretical/Quantum Chemistry
Laplace2d	261121	2343957	2.82e-05	5.05e-04	Numerical PDE
Laplace3d	1000000	6960000	3.29e-04	2.63e-03	Numerical PDE

Table 5.9: Performance comparison of Chebyshev LOBPCG with Chebyshev Davidson and LOBPCG ($n_w = 5$).

Matrix	Chebyshev Davidson			LOBPCG			Chebyshev LOBPCG		
	mvp	iter	CPU	mvp	iter	CPU	mvp	iter	CPU
Tre20k	61500	300	24.51	225	16	**0.43**	3370	15	1.34
finan512	5330	26	4.69	1185	80	8.55	4660	21	**4.46**
hb1138bus	866740	4228	6.68	1050	71	0.19	14550	67	**0.19**
cfd1	113980	556	143.94	52170	3479	411.22	53895	250	**80.09**
Muu	9840	48	0.93	930	63	0.58	4150	19	**0.52**
wathen100	7995	39	3.38	540	37	1.92	3800	17	**1.92**
obstclae	22140	108	8.63	3150	211	11.58	17130	79	**8.08**
shallowwater1	4920	24	**4.31**	4245	284	30.25	5050	23	5.10
minsurfo	36490	178	14.46	3585	240	12.31	20920	97	**9.63**
Ga41As41H72	121360	592	1169.04	35220	2349	1424.86	60560	281	**633.34**
GaAsH6	16400	80	30.59	22590	1507	176.87	11325	52	**24.17**
Ga3As3H12	22755	111	67.42	26235	1750	199.76	15800	73	**57.55**
Si87H76	7175	35	**45.54**	1770	119	62.14	6165	28	49.26
Ge99H100	7585	37	33.39	1935	130	36.83	6380	29	**33.30**
SiO2	5945	29	37.08	2295	154	58.65	3585	16	**28.38**
CO	2460	12	**12.36**	645	24	24.75	2080	9	17.45
Si41Ge41H72	7380	36	61.62	1635	110	**53.51**	5950	27	56.51
Laplace2d	112340	548	366.12	29355	1958	742.16	45900	213	**171.00**
Laplace3d	36115	233	405.13	1830	123	**215.20**	13440	81	225.72

Table 5.10: Performance comparison of Chebyshev LOBPCG with Chebyshev Davidson and LOBPCG ($n_w = 100$).

Matrix	Chebyshev Davidson			LOBPCG			Chebyshev LOBPCG		
	mvp	iter	CPU	mvp	iter	CPU	mvp	iter	CPU
Tre20k	178760	218	71.77	5520	93	**19.06**	55500	60	27.79
finan512	91840	112	**89.61**	20940	350	295.71	64920	71	90.79
hb1138bus	4754360	5798	40.77	10380	174	**1.46**	142360	161	1.92
cfd1	1174240	1432	1452.59	249060	4152	2952.31	526740	608	**881.34**
Muu	61500	75	6.87	10800	181	11.85	44200	47	**6.28**
wathen100	87740	107	35.77	9300	156	46.34	51160	55	**29.50**
obstclae	103320	126	42.01	22140	370	128.55	74420	82	**40.41**
shallowwater1	97580	119	**95.08**	70740	1180	959.62	72700	80	100.96
minsurfo	418200	510	170.05	34260	572	191.96	208580	238	**124.18**
Ga41As41H72	1110280	1354	9748.58	299820	4998	16288.36	445900	514	**4651.18**
GaAsH6	502660	613	827.67	166920	2783	1801.57	230940	264	**453.99**
Ga3As3H12	NA	NA	NA	NA	NA	NA	649760	751	**1867.57**
Si87H76	68880	84	414.92	16680	279	824.70	50340	54	**378.38**
Ge99H100	54940	67	216.29	13800	231	324.22	41740	44	**191.66**
SiO2	97580	119	544.97	25260	422	838.93	67300	74	**415.02**
CO	120540	147	573.39	25920	433	1081.17	75280	83	**448.55**
Si41Ge41H72	65600	80	465.04	16560	277	667.32	47720	51	**384.08**
Laplace2d	767520	936	2545.36	242760	4047	10345.83	374520	431	**1706.91**
Laplace3d	228160	368	3017.93	17220	288	3151.84	105180	155	**2342.18**

Chapter 6

Conclusions

In this dissertation, we study three important classes of large-scale and complex physical problems with special structures, i.e., computing the steady-state solutions of Allen-Cahn (AC) and Cahn–Hilliard (CH) equations, computing the ground states of Bose-Einstein condensation (BEC), and the Bethe-Salpeter eigenvalue problem (BSE). These problems can be addressed from the optimization perspective. We propose several new preconditioned conjugate gradient (PCG) methods to find the *local* or *global* minimizer of the corresponding objective function (energy functional or Rayleigh quotient), which takes advantage of the special structures and preserve the underlying physical properties. We show that our methods exhibit good convergence properties theoretically. We perform extensive numerical experiments to validate our methods and compare the proposed PCG methods with other existing methods. Our results clearly show that the new PCG methods are competitive or comparable with the state-of-art methods.

In Chapter 3, the preconditioned conjugate gradient (PCG) method and Picard iterative method with Anderson acceleration (AA) are proposed for computing the steady-state solutions of Allen-Cahn (AC) and Cahn-Hilliard (CH) equations. We show the Picard iterations are contractive near the desired solutions u^* with most elements are 1 in absolute value. In the meanwhile, we show the discrete energy stability and 2-norm stability of the ETD1 scheme for the CH equation. Numerical experiments were carried to compare the performances of our methods with the ETDRK2 scheme. We conclude that the PCG method is far more efficient than the other two methods for computing the steady-state solutions of the AC and CH equations.

In Chapter 4, we propose a preconditioned nonlinear conjugate gradient method to compute

the ground states of the GPE with fast rotation and large nonlinearities that arise in the modeling of Bose–Einstein Condensates in real arithmetic. We develop a problem-dependent Hessian preconditioner involving the rotational speed Ω, which is very efficient especially for solving BECs with high nonlinearity and high rotational speeds. Also, we provide an efficient method to perform fast energy functional evaluation without repeated computation in the original problem dimension. Exact line search can be enabled by fast energy evaluation at many different step sizes at little extra cost. Furthermore, our methodologies can be extended to solve other different types of BEC (e.g., spin-1 and rotating dipolar BEC) in the future.

In Chapter 5, we propose a variant of LOBPCG involving the preconditioned gradient and polynomial filtered Ritz vectors into the search subspace simultaneously, for computing a large number of the lowest eigenvalues of the algebraic product eigenvalue problem (1.25) and (1.32) arising from the Bethe-Salpeter equation. We construct an ideal method and establish the global quasi-optimality of this algorithm near convergence. We compared our algorithm with several widely known and adopted algorithms with incomplete Cholesky factorization preconditioners for the gradient and polynomial filters of relatively low degrees for the Ritz vectors. Numerical results have shown that our algorithm is competitive given a relatively low degree of polynomial filter and a preconditioner of low-to-modest quality. In the meanwhile, we extend our method to solve the large and sparse standard symmetric eigenvalue problems, and our numerical results demonstrate that our method are competitive or comparable with other existing methods.

In the future, our proposed PCG method can be employed to solve other different types of BEC. We use the 2-component as an example to briefly discuss a generalization of our ideas. The energy functional is

$$E_{\beta,\delta} = \Big[\phi_1^* L_s \phi_1 + \phi_2^* L_s \phi_2 + \delta \phi_1^* \phi_1 + \frac{\beta_{11}}{2}\phi_1^* \operatorname{diag}(|\phi_1|^2)\phi_1 + \frac{\beta_{22}}{2}\phi_2^* \operatorname{diag}(|\phi_2|^2)\phi_2 \qquad (6.1)$$
$$+\beta_{12}(\phi_1\phi_2)^*(\phi_1\phi_2) + 2\lambda \operatorname{Re}(\phi_1\bar{\phi}_2)\Big]\, h^2, \text{ with } \|\phi_1\|_{\ell^2} + \|\phi_2\|_{\ell^2} = 1,$$

where δ is the Raman transition constant, λ is the effective Rabi frequency describing the strength to realize the internal atomic Josephson junction by a Raman transition, and β_{ij} are interaction

constants between two components [16]. The gradient of $E_{\beta,\delta}$ is

$$\frac{\partial E_{\beta,\delta}}{\partial \phi} = \begin{pmatrix} (L_s + \delta I)\phi_1 + \mathrm{diag}(\beta_{11}|\phi_1|^2 + \beta_{12}|\phi_2|^2)\phi_1 + \lambda\phi_2 \\ L_s\phi_1 + \mathrm{diag}(\beta_{12}|\phi_1|^2 + \beta_{22}|\phi_2|^2)\phi_1 + \lambda\phi_1 \end{pmatrix} - E_{\beta,\delta} \begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix}. \tag{6.2}$$

From here, one can follow our strategies for solving rotational BEC in Chapter 4: write (6.1) and (6.2) in real arithmetic form, take the derivative of (6.2) with respect to ϕ to form the Hessian $H_{\beta,\delta}$. Similarly, our method can be applied to the spin-1 BEC, whose energy functional and gradient are known explicitly [94, 106].

Another possible project will focus on the eigen-solvers which are based on the polynomial filter. In Chapter 5, we point out that the quality of the polynomial filter depends on its degree and ratio between the the width of the wanted spectrum against the width of the unwanted spectrum (5.36). Higher degree may involve more computational cost. In practice, we always prefer a polynomial filter with low degree. Given that the degree of polynomial filter is fixed, the polynomial filter tends to become less effective when $\lambda_{\max}$ becomes larger. Suppose that the action of the inverse of the matrix can be performed efficiently, we can solve the eigen-problem based on the inverse of the matrix. Mostly, the ratio between the the width of the wanted spectrum against the width of the unwanted spectrum can be much larger, which only requires a low degree for the polynomial filter to be effective. For example, we can employ this strategy to solve the eigen-problem for the discrete Laplacian matrix, since its largest eigenvalue tends to $+\infty$ while the action of its inverse can be evaluated efficiently.

Appendix A

Supplementary materials

Proof of Theorem 4.2.1

Proof. The derivation follows from several applications of the product rule and quotient rule. More specifically, we have

$$\frac{\partial E(\phi)}{\partial \phi} = \frac{\partial \frac{\phi^T A \phi}{\phi^T \phi}}{\partial \phi} + \frac{\eta}{2h^d} \frac{\partial \frac{\phi^T B(\phi) \phi}{(\phi^T \phi)^2}}{\partial \phi}.$$

It is easy to obtain

$$\frac{\partial \frac{\phi^T A \phi}{\phi^T \phi}}{\partial \phi} = \frac{2}{\phi^T \phi} \left(A\phi - \frac{\phi^T A \phi}{\phi^T \phi} \phi \right), \tag{A.1}$$

and

$$\frac{\partial \frac{\phi^T B(\phi) \phi}{(\phi^T \phi)^2}}{\partial \phi} = \frac{1}{(\phi^T \phi)^2} \left(\frac{\partial \phi^T B(\phi) \phi}{\partial \phi} \right) - 4 \frac{\phi^T B(\phi) \phi}{(\phi^T \phi)^3} \phi. \tag{A.2}$$

From [58, Theorem 5.1], we have

$$\begin{aligned} \frac{\partial B(\phi)\phi}{\partial \phi} &= B(\phi) + 2\mathrm{diag}(\phi)^2 + 2 \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix} \mathrm{diag}(\phi) \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix} \mathrm{diag}(\phi) \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix} \\ &= B(\phi) + 2 \begin{pmatrix} \mathrm{diag}(\phi_r^2) & \mathrm{diag}(\phi_r \phi_g) \\ \mathrm{diag}(\phi_r \phi_g) & \mathrm{diag}(\phi_g^2) \end{pmatrix} \end{aligned}$$

Then, it follows that

$$\frac{\partial \phi^T B(\phi)\phi}{\partial \phi} = B(\phi)\phi + \left(\frac{\partial B(\phi)\phi}{\partial \phi}\right)^T \phi \tag{A.3}$$
$$= B(\phi)\phi + \left(B(\phi) + 2\begin{pmatrix} \mathrm{diag}(\phi_r^2) & \mathrm{diag}(\phi_r\phi_g) \\ \mathrm{diag}(\phi_r\phi_g) & \mathrm{diag}(\phi_g^2) \end{pmatrix}\right)^T \phi = 4B(\phi)\phi.$$

It follows from (A.2),

$$\frac{\eta}{2h^d}\frac{\partial \frac{\phi^T B(\phi)\phi}{(\phi^T\phi)^2}}{\partial \phi} = \frac{2\eta}{h^d}\left(\frac{B(\phi)\phi}{(\phi^T\phi)^2} - \frac{\phi^T B(\phi)\phi}{(\phi^T\phi)^3}\phi\right) \tag{A.4}$$

Combining (A.1) and (A.4), we get

$$\frac{\partial E(\phi)}{\partial \phi} = \frac{2}{\phi^T\phi}\left(A\phi - \frac{\phi^T A\phi}{\phi^T\phi}\phi\right) + \frac{2\eta}{h^d}\left(\frac{B(\phi)\phi}{(\phi^T\phi)^2} - \frac{\phi^T B(\phi)\phi}{(\phi^T\phi)^3}\phi\right) \tag{A.5}$$
$$= \frac{2}{\phi^T\phi}\left(A\phi + \eta\frac{B(\phi)\phi}{h^d\phi^T\phi} - \frac{\phi^T A\phi}{\phi^T\phi}\phi - \eta\frac{\phi^T B(\phi)\phi}{h^d(\phi^T\phi)^2}\phi\right) = \frac{2}{\phi^T\phi}(A(\phi)\phi - \lambda(\phi)\phi).$$

Next, we know

$$\frac{\partial^2 E(\phi)}{\partial \phi^2} = \frac{2}{(\phi^T\phi)^2}\left(\phi^T\phi\frac{\partial(A(\phi)\phi - \lambda(\phi)\phi)}{\partial\phi} - 2(A(\phi)\phi - \lambda(\phi)\phi)\phi^T\right) \tag{A.6}$$
$$= \frac{2}{\phi^T\phi}\left(\frac{\partial(A(\phi)\phi - \lambda(\phi)\phi)}{\partial\phi} - \frac{2(A(\phi)\phi - \lambda(\phi)\phi)\phi^T}{\phi^T\phi}\right)$$

Again, from [58, Theorem 5.1], we have

$$\frac{\partial A(\phi)\phi}{\partial\phi} = A + \frac{\eta}{h^d\phi^T\phi}\left[\begin{pmatrix} \mathrm{diag}(3\phi_r^2 + \phi_g^2) & 2\mathrm{diag}(\phi_r\phi_g) \\ 2\mathrm{diag}(\phi_r\phi_g) & \mathrm{diag}(\phi_r^2 + 3\phi_g^2) \end{pmatrix} - \frac{2}{\phi^T\phi}B(\phi)\phi\phi^T\right] \tag{A.7}$$

Also, we have

$$\frac{\partial \lambda(\phi)\phi}{\partial\phi} = \phi(\frac{\partial\lambda(\phi)}{\partial\phi})^T + \lambda(\phi)I \tag{A.8}$$
$$= \frac{2}{\phi^T\phi}\phi\left(\phi^T A + 2\eta\frac{\phi^T B(\phi)}{h^d\phi^T\phi} - \phi^T\frac{\phi^T A\phi}{\phi^T\phi} - 2\eta\phi^T\frac{\phi^T B(\phi)\phi}{h^d(\phi^T\phi)^2}\right) + \lambda(\phi)I$$

Combine (A.7) and (A.8), we get

$$\begin{aligned}
&\frac{\partial(A(\phi)\phi-\lambda(\phi)\phi)}{\partial\phi} \\
&= A+\frac{\eta}{h^d\phi^T\phi}\left[\begin{pmatrix}\mathrm{diag}(3\phi_r^2+\phi_g^2) & 2\mathrm{diag}(\phi_r\phi_g)\\ 2\mathrm{diag}(\phi_r\phi_g) & \mathrm{diag}(\phi_r^2+3\phi_g^2)\end{pmatrix}-\frac{2}{\phi^T\phi}B(\phi)\phi\phi^T\right] \\
&\quad-\frac{2}{\phi^T\phi}\phi\left(\phi^TA+2\eta\frac{\phi^TB(\phi)}{h^d\phi^T\phi}-\phi^T\frac{\phi^TA\phi}{\phi^T\phi}-2\eta\phi^T\frac{\phi^TB(\phi)\phi}{h^d(\phi^T\phi)^2}\right)-\lambda(\phi)I \\
&= A+\frac{\eta}{h^d\phi^T\phi}\begin{pmatrix}\mathrm{diag}(3\phi_r^2+\phi_g^2) & 2\mathrm{diag}(\phi_r\phi_g)\\ 2\mathrm{diag}(\phi_r\phi_g) & \mathrm{diag}(\phi_r^2+3\phi_g^2)\end{pmatrix}-\lambda(\phi)I \\
&\quad-2\eta\frac{B(\phi)}{h^d\phi^T\phi}\frac{\phi\phi^T}{\phi^T\phi}-2\frac{\phi\phi^T}{\phi^T\phi}A-4\eta\frac{\phi\phi^T}{\phi^T\phi}\frac{B(\phi)}{h^d\phi^T\phi}+2\frac{\phi^TA\phi}{\phi^T\phi}\frac{\phi\phi^T}{\phi^T\phi}+4\eta\frac{\phi\phi^T}{\phi^T\phi}\frac{\phi^TB(\phi)\phi}{h^d(\phi^T\phi)^2}
\end{aligned} \tag{A.9}$$

Also, we have

$$\begin{aligned}
&\frac{2(A(\phi)\phi-\lambda(\phi)\phi)\phi^T}{\phi^T\phi} \\
&= \frac{2\left((A+\frac{\eta}{h^d}\frac{B(\phi)}{\phi^T\phi})\phi-(\frac{\phi^TA\phi}{\phi^T\phi}+\frac{\eta}{h^d}\frac{\phi^TB(\phi)\phi}{(\phi^T\phi)^2})\phi\right)\phi^T}{\phi^T\phi} \\
&= 2A\frac{\phi\phi^T}{\phi^T\phi}+2\eta\frac{B(\phi)}{h^d\phi^T\phi}\frac{\phi\phi^T}{\phi^T\phi}-2\frac{\phi^TA\phi}{\phi^T\phi}\frac{\phi\phi^T}{\phi^T\phi}-2\eta\frac{\phi^TB(\phi)\phi}{h^d(\phi^T\phi)^2}\frac{\phi\phi^T}{\phi^T\phi}
\end{aligned} \tag{A.10}$$

Combining (A.9) and (A.10), it follows from (A.6) that

$$\begin{aligned}
\frac{\partial^2E(\phi)}{\partial\phi^2} = \frac{2}{\phi^T\phi}\Bigg\{&A+\frac{\eta}{h^d\phi^T\phi}\begin{pmatrix}\mathrm{diag}(3\phi_r^2+\phi_g^2) & 2\mathrm{diag}(\phi_r\phi_g)\\ 2\mathrm{diag}(\phi_r\phi_g) & \mathrm{diag}(\phi_r^2+3\phi_g^2)\end{pmatrix}-\lambda(\phi)I \\
&-2A\frac{\phi\phi^T}{\phi^T\phi}-2\frac{\phi\phi^T}{\phi^T\phi}A-4\eta\frac{B(\phi)}{h^d\phi^T\phi}\frac{\phi\phi^T}{\phi^T\phi}-4\eta\frac{\phi\phi^T}{\phi^T\phi}\frac{B(\phi)}{h^d\phi^T\phi} \\
&+4\frac{\phi^TA\phi}{\phi^T\phi}\frac{\phi\phi^T}{\phi^T\phi}+6\eta\frac{\phi\phi^T}{\phi^T\phi}\frac{\phi^TB(\phi)\phi}{h^d(\phi^T\phi)^2}\Bigg\}.
\end{aligned}$$

Next. we will show that ϕ and $\widehat{\phi}$ are the eigenvectors of $\frac{\partial^2E(\phi)}{\partial\phi^2}$ associated with the zero

eigenvalue. Note that $B(\phi)\phi = \begin{pmatrix} \phi_r^3 + \phi_r\phi_g^2 \\ \phi_r^2\phi_g + \phi_g^3 \end{pmatrix} \in \mathbb{R}^{2n}$. Then, we have

$$\begin{aligned}
\frac{\partial^2 E(\phi)}{\partial \phi^2}\phi = \frac{2}{\phi^T\phi}\Bigg\{ & A\phi\frac{\eta}{h^d\phi^T\phi}\begin{pmatrix} 3\phi_r^3 + 3\phi_r\phi_g^2 \\ 3\phi_r^2\phi_g + 3\phi_g^3 \end{pmatrix} - \lambda(\phi)\phi - 2A\phi - 2\frac{\phi^T A\phi}{\phi^T\phi}\phi \\
& - \frac{4\eta}{h^d\phi^T\phi}\begin{pmatrix} \phi_r^3 + \phi_r\phi_g^2 \\ \phi_r^2\phi_g + \phi_g^3 \end{pmatrix} - \frac{4\eta}{h^d\phi^T\phi}\frac{\phi^T B(\phi)\phi}{\phi^T\phi}\phi \\
& + 4\frac{\phi^T A\phi}{\phi^T\phi}\phi + \frac{6\eta}{h^d\phi^T\phi}\frac{\phi^T B(\phi)\phi}{\phi^T\phi}\phi\Bigg\} \\
= \frac{2}{\phi^T\phi}\Bigg\{ & -A\phi - \frac{\eta}{h^d\phi^T\phi}\begin{pmatrix} \phi_r^3 + \phi_r\phi_g^2 \\ \phi_r^2\phi_g + \phi_g^3 \end{pmatrix} + 2\frac{\phi^T A\phi}{\phi^T\phi}\phi \\
& + \frac{2\eta}{h^d\phi^T\phi}\frac{\phi^T B(\phi)\phi}{\phi^T\phi}\phi - \lambda(\phi)\phi\Bigg\} \\
= \frac{2}{\phi^T\phi}\Bigg\{ & -A\phi - \frac{\eta}{h^d\phi^T\phi}B(\phi)\phi + \lambda(\phi)\phi\Bigg\} = \frac{2}{\phi^T\phi}\left\{-\left(A(\phi)\phi - \lambda(\phi)\phi\right)\right\},
\end{aligned}$$

which is zero since $\frac{\partial E(\phi)}{\partial \phi} = \frac{2}{\phi^T\phi}\left(A(\phi)\phi - \lambda(\phi)\phi\right) = 0$, i.e., ϕ is a stationary point of $E(\phi)$ (local or global minimum, or saddle point). On the other hand, it is easy to see that $\phi^T\widehat{\phi} = 0$, then we have

$$\phi^T A\widehat{\phi} = -\phi_r^T L_s\phi_g + \Omega\phi_r^T L_\omega\phi_r + \Omega\phi_g^T L_\omega\phi_g + \phi_g^T L_s\phi_r = 0, \tag{A.11}$$

since L_s is symmetric and L_ω is skew-symmetric such that $u^T L_\omega u = 0$ for any $u \in \mathbb{R}^n$. Also, we have

$$\phi^T B(\phi)\widehat{\phi} = -(\phi_r^3)^T\phi_g - \phi_r^T\phi_g^3 + \phi_g^T\phi_r^3 + (\phi_g^3)^T\phi_r = 0. \tag{A.12}$$

Combining (A.11) and (A.12), we can easily obtain

$$\begin{aligned}
\frac{\partial^2 E}{\partial \phi^2}\widehat{\phi} &= \frac{2}{\phi^T\phi}\left\{A\widehat{\phi} + \frac{\eta}{h^d\phi^T\phi}\begin{pmatrix}-\phi_r^2\phi_g - \phi_g^3\\ \phi_r\phi_g^2 + \phi_r^3\end{pmatrix} - \lambda(\phi)\widehat{\phi}\right\}\\
&= \frac{2}{\phi^T\phi}\left\{\begin{pmatrix}-L_s\phi_g + \Omega L_\omega\phi_r\\ L_s\phi_r + \Omega L_\omega\phi_g\end{pmatrix} + \frac{\eta}{h^d\phi^T\phi}\begin{pmatrix}-\phi_r^2\phi_g - \phi_g^3\\ \phi_r\phi_g^2 + \phi_r^3\end{pmatrix} - \lambda(\phi)\begin{pmatrix}-\phi_g\\ \phi_r\end{pmatrix}\right\}\\
&= \frac{2}{\phi^T\phi}\begin{pmatrix}0 & -I\\ I & 0\end{pmatrix}\left\{\begin{pmatrix}L_s\phi_r + \Omega L_\omega\phi_g\\ L_s\phi_g - \Omega L_\omega\phi_r\end{pmatrix} + \frac{\eta}{h^d\phi^T\phi}\begin{pmatrix}\phi_r^3 + \phi_r\phi_g^2\\ \phi_r^2\phi_g + \phi_g^3\end{pmatrix} - \lambda(\phi)\begin{pmatrix}\phi_r\\ \phi_g\end{pmatrix}\right\}\\
&= \begin{pmatrix}0 & -I\\ I & 0\end{pmatrix}(A(\phi)\phi - \lambda(\phi)\phi) = \begin{pmatrix}0 & I\\ -I & 0\end{pmatrix}\frac{\partial^2 E(\phi)}{\partial\phi^2}\phi.
\end{aligned}$$

Therefore, if ϕ is a stationary point of $E(\phi)$ such that $\frac{\partial^2 E}{\partial\phi^2}\phi = A(\phi)\phi - \lambda(\phi)\phi = 0$, we also have $\frac{\partial^2 E}{\partial\phi^2}\widehat{\phi} = 0$. □

Proof of Theorem 4.4.1

Proof. First, it is easy to obtain that $P\phi = \phi^T P = 0$, then we have

$$PL^{-1}P\frac{\partial^2 E(\phi)}{\partial\phi^2}PL^{-T}P = 2h^d PL^{-1}PH_pPL^{-T}P. \tag{A.13}$$

Since $P = I - h^d WW^T$, we have

$$\begin{aligned}
&PL^{-1}PH_pPL^{-T} \qquad (A.14)\\
&= L^{-1}H_pL^{-T} - h^dL^{-1}H_pWW^TL^{-T} - h^dL^{-1}WW^TH_pL^{-T}\\
&\quad + h^{2d}L^{-1}WW^TH_pWW^TL^{-T} - h^dWW^TL^{-1}H_pL^{-T}\\
&\quad + h^{2d}WW^TL^{-1}H_pWW^TL^{-T} + h^{2d}WW^TL^{-1}WW^TH_pL^{-T}\\
&\quad - h^{3d}WW^TL^{-1}WW^TH_pWW^TL^{-T},
\end{aligned}$$

and

$$
\begin{aligned}
& PL^{-1}PH_pPL^{-T}h^dWW^T \qquad (A.15)\\
=\ & h^dL^{-1}H_pL^{-T}WW^T - h^{2d}L^{-1}H_pWW^TL^{-T}WW^T\\
& - h^{2d}L^{-1}WW^TH_pL^{-T}WW^T + h^{3d}L^{-1}WW^TH_pWW^TL^{-T}WW^T\\
& - h^{2d}WW^TL^{-1}H_pL^{-T}WW^T + h^{3d}WW^TL^{-1}H_pWW^TL^{-T}WW^T\\
& + h^{3d}WW^TL^{-1}WW^TH_pL^{-T}WW^T - h^{4d}WW^TL^{-1}WW^TH_pWW^TL^{-T}WW^T.
\end{aligned}
$$

Then, we obtain

$$
\begin{aligned}
& PL^{-1}PH_pPL^{-T}P\\
=\ & L^{-1}H_pL^{-T} + W\left(h^{2d}W^TL^{-1}H_pWW^TL^{-T} - h^dW^TL^{-1}H_pL^{-T}\right.\\
& +h^{2d}W^TL^{-1}WW^TH_pL^{-T} - h^{3d}W^TL^{-1}WW^TH_pWW^TL^{-T}\\
& +h^{2d}W^TL^{-1}H_pL^{-T}WW^T - h^{3d}W^TL^{-1}H_pWW^TL^{-T}WW^T\\
& \left.-h^{3d}W^TL^{-1}WW^TH_pL^{-T}WW^T + h^{4d}W^TL^{-1}WW^TH_pWW^TL^{-T}WW^T\right)\\
& + L^{-1}W\left(h^{2d}W^TH_pWW^TL^{-T} - h^dW^TH_pL^{-T} + h^{2d}W^TH_pL^{-T}WW^T\right.\\
& \left.-h^{3d}W^TH_pWW^TL^{-T}WW^T\right) + L^{-1}H_pW\left(h^{2d}W^TL^{-T}WW^T - h^dW^TL^{-T}\right)\\
& - h^dL^{-1}H_pL^{-T}WW^T.
\end{aligned}
$$

Therefore, $PL^{-1}PH_pPL^{-T}P$ is a rank-8 update of $L^{-1}H_pL^{-T}$. Also, W_1, W_2 and $W_3 \in \mathbb{R}^{2n\times 2}$ can be obtained based on the above expression, respectively.

On the other hand, we have $L^{-1}H_pL^{-T} = L^{-1}(H_p + \sigma I - \sigma I)L^{-T} = I - \sigma L^{-1}L^{-T}$. Then,

$h^d L^{-1} H_p L^{-T} W W^T = h^d (I - \sigma L^{-1} L^{-T}) W W^T$. If $\sigma = 0$, then $L^{-1} H_p L^{-T} = I$ and we have

$$
\begin{aligned}
& P L^{-1} P H_p P L^{-T} P \\
= & I + W \big(h^{2d} W^T L^T W W^T L^{-T} - h^d W^T L^T L^{-T} + h^{2d} W^T L^{-1} W W^T L \\
& - h^{3d} W^T L^{-1} W W^T H_p W W^T L^{-T} + h^{2d} W^T - h^{3d} W^T L^{-1} H_p W W^T L^{-T} W W^T \\
& - h^{3d} W^T L^{-1} W W^T L W W^T + h^{4d} W^T L^{-1} W W^T H_p W W^T L^{-T} W W^T - h^d W^T \big) \\
& + L^{-1} W \big(h^{2d} W^T H_p W W^T L^{-T} - h^d W^T L + h^{2d} W^T L W W^T \\
& - h^{3d} W^T H_p W W^T L^{-T} W W^T \big) + L^{-1} H_p W \big(h^{2d} W^T L^{-T} W W^T - h^d W^T L^{-T} \big)
\end{aligned}
$$

Therefore, $P L^{-1} P H_p P L^{-T} P$ is a rank-6 update of I. □

Proof of Lemma 5.3.3

Proof. Given the decomposition (5.32), the normalization $\|x_k\|_M = 1$ leads to

$$
\begin{aligned}
\|x_k\|_M^2 &= (\beta_{0k} F^m x_0 + \gamma_{0k} g_{0k})^T M (\beta_{0k} F^m x_0 + \gamma_{0k} g_{0k}) \\
&= \|F^m x_0\|_M^2 \beta_{0k}^2 + 2 (g_{0k}^T M F^m x_0) \gamma_{0k} \beta_{0k} + \|g_{0k}\|_M^2 \gamma_{0k}^2 \\
&= \beta_{0k}^2 + 2 \mu_{0k} \gamma_{0k} \beta_{0k} + \gamma_{0k}^2 = (\gamma_{0k} + \mu_{0k} \beta_{0k})^2 + (1 - \mu_{0k}^2) \beta_{0k}^2 = 1,
\end{aligned} \tag{A.16}
$$

where $\mu_{0k} = g_{0k}^T M F^m x_0 = (F^m x_0, g_{0k})_M \in (-1, 1)$, since $\|F^m x_0\|_M = \|g_{0k}\|_M = 1$.

Let $r_0 = MKMF^m x_0 - \rho(F^m x_0) M F^m x_0$ be the eigenresidual associated with $F^m x_0$ and define

$d_{0k} = \rho(g_{0k}) - \rho(F^m x_0)$. The Rayleigh quotient of x_k is therefore

$$\begin{aligned}
\rho(x_k) &= \frac{x_k^T MKMx_k}{x_k^T Mx_k} = x_k^T MKMx_k \\
&= (\beta_{0k} F^m x_0 + \gamma_{0k} g_{0k})^T MKM(\beta_{0k} F^m x_0 + \gamma_{0k} g_{0k}) \\
&= ((F^m x_0)^T MKMF^m x_0)\beta_{0k}^2 + 2(g_{0k}^T MKMF^m x_0)\gamma_{0k}\beta_{0k} \\
&\quad + (g_{0k}^T MKMg_{0k})\gamma_{0k}^2 \\
&= \rho(F^m x_0)\beta_{0k}^2 + 2g_{0k}^T(r_0 + \rho(F^m x_0)MF^m x_0)\gamma_{0k}\beta_{0k} + \rho(g_{0k})\gamma_{0k}^2 \\
&= \rho(F^m x_0)\beta_{0k}^2 + 2\rho(F^m x_0)(g_{0k}^T MF^m x_0)\gamma_{0k}\beta_{0k} + 2(g_{0k}^T r_0)\gamma_{0k}\beta_{0k} \\
&\quad + (\rho(F^m x_0) + d_{0k})\gamma_{0k}^2 \\
&= \rho(F^m x_0)(\beta_{0k}^2 + 2\mu_{0k}\gamma_{0k}\beta_{0k} + \gamma_{0k}^2) + 2(g_{0k}^T r_0)\gamma_{0k}\beta_{0k} + d_{0k}\gamma_{0k}^2 \\
&= \rho(F^m x_0) + 2(g_{0k}^T r_0)\gamma_{0k}\beta_{0k} + d_{0k}\gamma_{0k}^2.
\end{aligned}$$

Then, we have

$$2(g_{0k}^T r_0)\gamma_{0k}\beta_{0k} + d_{0k}\gamma_{0k}^2 = \rho(x_k) - \rho(F^m x_0).$$

Due to (5.25), we have $\rho(F^m x_0) \leq \rho(x_0)$, it follows that

$$(\rho(x_k) - \lambda_1) - (\rho(x_0) - \lambda_1) = \rho(x_k) - \rho(x_0) \leq \rho(x_k) - \rho(F^m x_0) \leq \rho(x_k) - \lambda_1.$$

Using Assumption 5.3.1, we have

$$-(1 - \underline{\xi}^k)(\rho(x_0) - \lambda_1) \leq 2(g_{0k}^T r_0)\gamma_{0k}\beta_{0k} + d_{0k}\gamma_{0k}^2 \leq \overline{\xi}^k(\rho(x_0) - \lambda_1).$$

By (2.3), we divide the above inequality by $\sin^2\theta_0$ and obtain

$$-(1 - \underline{\xi}^k)(\rho(f_0) - \lambda_1) \leq 2\frac{g_{0k}^T r_0}{\sin\theta_0}\left(\frac{\gamma_{0k}}{\sin\theta_0}\right)\beta_{0k} + d_{0k}\left(\frac{\gamma_{0k}}{\sin\theta_0}\right)^2 \leq \overline{\xi}^k(\rho(f_0) - \lambda_1)$$

Note that the left-hand side and right-hand side of the above inequality are both $\mathcal{O}(1)$ independent of θ_0. Also, note that $\frac{|g_{0k}^T r_0|}{\sin\theta_0} \leq \frac{\|g_{0k}\|\|r_0\|}{\sin\theta_0} = \mathcal{O}(1)$, where $\|r_0\| = \mathcal{O}(\sin\theta_{F^m x_0}) = \mathcal{O}(\sin\theta_{x_0})$ by (5.26). Then, by the above inequality, we have that $\gamma_{0k} = \mathcal{O}(\sin^q\theta_0)$ for some scalar $q \geq 1$. Remind that

$\mu_{0k} = g_{0k}^T M F^m x_0 = (F^m x_0, g_{0k})_M \in (-1, 1)$ from (A.16). Consequently, we have

$$\begin{aligned}\beta_{0k} &= -\mu_{0k}\gamma_{0k} + \sqrt{1 - \gamma_{0k}^2 + \mu_{0k}^2\gamma_{0k}^2} = -\mu_{0k}\gamma_{0k} + 1 - \frac{1}{2}(1 - \mu_{0k}^2)\gamma_{0k}^2 + \mathcal{O}(\gamma_{0k}^4) \\ &= 1 - \mathcal{O}(\sin^q \theta_0)\end{aligned}$$

□

Proof of Theorem 5.3.1

Proof. Since $y = \beta_y F^m x_0 + \gamma_y g_y, g_y \in W_k$, $\|y\|_M = \|F^m x_0\|_M = \|g_y\|_M = 1$, from Lemma 5.3.3, we have $\gamma_y = \mathcal{O}(\sin^q \theta_0)$ and $|1 - \beta_y| = \mathcal{O}(\sin^q \theta_0)$ for some scalar $q \geq 1$. Similarly, the iterate $x_k = \beta_{0k} F^m x_0 + \gamma_{0k} g_{0k}$ satisfies $\gamma_{0k} = \mathcal{O}(\sin^q \theta_0)$ and $|1 - \beta_{0k}| = \mathcal{O}(\sin^q \theta_0)$. Since $\rho(F^m x_0) \leq \rho(x_0)$ by Assumption 5.3.2, we have $\rho(F^m x_0) - \rho(x_k) \leq \rho(x_0) - \lambda_1 = \mathcal{O}(\sin^2 \theta_0)$. Similarly, we have $\rho(F^m x_0) - \rho(y) = \mathcal{O}(\sin^2 \theta_0)$. Also, for a fixed m, we have $(MKM - \rho(F^m x_0)M)F^m x_0 = \mathcal{O}(\sin \theta_{F^m x_0}) = \mathcal{O}(\sin \theta_0)$ by (2.4) and (5.26). It follows that

$$\begin{aligned}&(MKM - \rho(y)M)y - (MKM - \rho(x_k)M)x_k \qquad (A.17)\\ &= (MKM - \rho(F^m x_0)M)(\beta_y F^m x_0 + \gamma_y g_y) \\ &\quad - (MKM - \rho(F^m x_0)M)(\beta_{0k} F^m x_0 + \gamma_{0k} g_{0k}) \\ &\quad + (\rho(F^m x_0) - \rho(y))My - (\rho(F^m x_0) - \rho(x_k))Mx_k \\ &= (\beta_y - 1 + 1 - \beta_{0k})(MKM - \rho(F^m x_0)M)F^m x_0 \\ &\quad + (MKM - \rho(F^m x_0)M)(\gamma_y g_y - \gamma_{0k} g_{0k}) + \mathcal{O}(\sin^2 \theta_0) \\ &= \big(\mathcal{O}(\sin^q \theta_0) + \mathcal{O}(\sin^q \theta_0)\big)\mathcal{O}(\sin \theta_0) \\ &\quad + (MKM - \rho(F^m x_0)M)(\gamma_y g_y - \gamma_{0k} g_{0k}) + \mathcal{O}(\sin^2 \theta_0) \\ &= (MKM - \rho(x_k)M)(\gamma_y g_y - \gamma_{0k} g_{0k}) \\ &\quad + (\rho(x_k) - \rho(F^m x_0))M(\gamma_y g_y - \gamma_{0k} g_{0k}) + \mathcal{O}(\sin^2 \theta_0) \\ &= \gamma_y (MKM - \rho(x_k)M)g_y - \gamma_{0k}(MKM - \rho(x_k)M)g_{0k} + \mathcal{O}(\sin^2 \theta_0).\end{aligned}$$

Note that $g_y, g_{0k} \in W_k = \text{span}\{p_0, p_1, \cdots, p_{k-1}\}$. Due to Proposition 5.3.1, we have $p_k^T(MKM - \rho(x_k)M)g_y = 0$ and $p_k^T(MKM - \rho(x_k)M)g_{0k} = 0$. It follows from (A.17) that

$$\begin{aligned}
&p_k^T\big((MKM - \rho(y)M)y - (MKM - \rho(x_k)M)x_k\big) \\
&= \gamma_y p_k^T(MKM - \rho(x_k)M)g_y - \gamma_{0k} p_k^T(MKM - \rho(x_k)M)g_{0k} + \mathcal{O}(\sin^2\theta_0) \\
&= \mathcal{O}(\sin^2\theta_0),
\end{aligned}$$

which is crucial for the rest of the proof, or equivalently

$$p_k^T(MKM - \rho(y)M)y = p_k^T(MKM - \rho(x_k)M)x_k + \mathcal{O}(\sin^2\theta_0). \tag{A.18}$$

In addition, note that $x_k - y = (\beta_{0k} - \beta_y)F^m x_0 + (\gamma_{0k} g_{0k} - \gamma_y g_y) = \mathcal{O}(\sin^q\theta_0)F^m x_0 + g_{0k}\mathcal{O}(\sin^q\theta_0) - g_y\mathcal{O}(\sin^q\theta_0) = \mathcal{O}(\sin^q\theta_0)$ and hence $My = Mx_k + \mathcal{O}(\sin^q\theta_0)$.

Let $R_3(\rho(x), \alpha p) = \frac{1}{2}\rho(x+\alpha p) - \big(\frac{1}{2}\rho(x) + \alpha p^T\nabla\frac{1}{2}\rho(x) + \frac{1}{2}\alpha^2 p^T\nabla^2\frac{1}{2}\rho(x)p\big)$ be the remainder of 2nd order Taylor expansion of $\frac{1}{2}\rho(x+\alpha p)$ at x with $\|x\|_M = 1$. Then,

$$\begin{aligned}
&\frac{1}{2}\rho(z) = \frac{1}{2}\rho(y+\alpha p_k) = \frac{1}{2}\rho(y) + \alpha p_k^T\frac{1}{2}\nabla\rho(y) + \frac{1}{2}\alpha^2 p_k^T\frac{1}{2}\nabla^2\rho(y)p_k + R_3(\rho(y), \alpha p_k) \qquad \text{(A.19)} \\
&= \frac{1}{2}\rho(y) + \alpha p_k^T(MKM - \rho(y)M)y + \frac{1}{2}\alpha^2 p_k^T\big((MKM - \rho(y)M) \\
&\quad - 2(MKMy - \rho(y)My)(My)^T - 2My(MKMy - \rho(y)My)^T\big)p_k + R_3(\rho(y), \alpha p_k) \\
&= \frac{1}{2}\rho(y) + \alpha\big(p_k^T(MKM - \rho(x_k)M)x_k + \mathcal{O}(\sin^2\theta_0)\big) \qquad \text{by (A.18)} \\
&\quad + \frac{1}{2}\alpha^2\big\{p_k^T\big((MKM - \rho(x_k)M) + (\rho(x_k) - \rho(y))M\big)p_k \\
&\quad - 2\big(p_k^T(MKM - \rho(x_k)M)x_k + \mathcal{O}(\sin\theta_0^2)\big)\big(x_k^T M + \mathcal{O}(\sin^q\theta_0)\big)p_k \\
&\quad -2p_k^T\big(Mx_k + \mathcal{O}(\sin^q\theta_0)\big)\big((x_k)^T(MKM - \rho(x_k)M)p_k + \mathcal{O}(\sin^2\theta_0)\big)\big\} + R_3(\rho(y), \alpha p_k) \\
&= \frac{1}{2}\rho(y) + \alpha p_k^T\frac{1}{2}\nabla\rho(x_k) + \alpha\mathcal{O}(\sin^2\theta_0) + \frac{1}{2}\alpha^2 p_k^T\{(MKM - \rho(x_k)M) \\
&\quad - 2r_k(Mx_k)^T - 2Mx_k r_k^T\}p_k + \alpha^2\big(\mathcal{O}(\sin^2\theta_0) + \|r_k\|\mathcal{O}(\sin^q\theta_0)\big) + R_3(\rho(y), \alpha p_k) \\
&= \frac{1}{2}\rho(x_k + \alpha p_k) + \frac{1}{2}(\rho(y) - \rho(x_k)) + R_3(\rho(y), \alpha p_k) \\
&\quad - R_3(\rho(x_k), \alpha p_k) + \mathcal{O}(\alpha\sin^2\theta_0) + \mathcal{O}(\alpha^2\sin^2\theta_0).
\end{aligned}$$

Let the global minimizer in U_{k+1} be $y_{k+1}^* = y(y_{k+1}^*) + \alpha(y_{k+1}^*)p_k$, with $y(y_{k+1}^*) \in U_k$, and

$\|y(y^*_{k+1})\|_M = 1$. Consider the decomposition $y(y^*_{k+1}) = v_1 \cos\theta_{y(y^*_{k+1})} + f_{y(y^*_{k+1})} \sin\theta_{y(y^*_{k+1})}$ with $f_{y(y^*_{k+1})} \perp Mv_1$ and $\|f_{y(y^*_{k+1})}\|_M = 1$ such that $\rho(y(y^*_{k+1})) - \lambda_1 = \mathcal{O}(\sin^2\theta_{y(y^*_{k+1})})$. Here, $\alpha(y^*_{k+1})$ is the optimal step size moving from $y(y^*_{k+1})$ in the direction of p_k, due to the global optimality of y^*_{k+1} in U_{k+1}. Under Assumption 5.3.3, it follows from Lemma 5.3.2 that

$$\rho(y^*_{k+1}) - \rho(y(y^*_{k+1})) = -\mathcal{O}(\sin^2\theta_{y(y^*_{k+1})})\mathcal{O}(\cos^2\angle(p_k, \nabla\rho(y(y^*_{k+1})))).$$

Then, we have

$$\begin{aligned}
&\rho(y^*_{k+1}) - \lambda_1 = [\rho(y^*_{k+1}) - \rho(y(y^*_{k+1}))] + [\rho(y(y^*_{k+1})) - \lambda_1] \\
&= -\mathcal{O}(\sin^2\theta_{y(y^*_{k+1})})\mathcal{O}(\cos^2\angle(p_k, \nabla\rho(y(y^*_{k+1})))) + \mathcal{O}(\sin^2\theta_{y(y^*_{k+1})}) \\
&= \mathcal{O}(\sin^2\theta_{y(y^*_{k+1})}).
\end{aligned}$$

On the other hand, as y^*_{k+1} is the global minimizer in U_{k+1}, we have

$$\rho(y^*_{k+1}) - \lambda_1 \le \rho(x_{k+1}) - \lambda_1 = \mathcal{O}(\sin^2\theta_{k+1}).$$

Therefore, we obtain that

$$\mathcal{O}(\sin\theta_{y(y^*_{k+1})}) \le \mathcal{O}(\sin\theta_{k+1}).$$

Given $y^*_{k+1} = y(y^*_{k+1}) + \alpha(y^*_{k+1})p_k$, by triangle inequality we have

$$\|y^*_{k+1}\|_M \le \|y(y^*_{k+1})\|_M + |\alpha(y^*_{k+1})|\|p_k\|_M = 1 + |\alpha(y^*_{k+1})|.$$

Also, the M-normalized y^*_{k+1} is $\frac{y^*_{k+1}}{\|y^*_{k+1}\|_M} = \frac{1}{\|y^*_{k+1}\|_M} y(y^*_{k+1}) + \frac{\alpha(y^*_{k+1})}{\|y^*_{k+1}\|_M} p_k$, and we can follow the proof of Lemma 5.3.3 to show that $\frac{\alpha(y^*_{k+1})}{\|y^*_{k+1}\|_M} = \mathcal{O}(\sin\theta_{y(y^*_{k+1})}) \le \mathcal{O}(\sin\theta_{k+1})$. Then, we could obtain that, $\frac{|\alpha(y^*_{k+1})|}{1+|\alpha(y^*_{k+1})|} \le \frac{|\alpha(y^*_{k+1})|}{\|y^*_{k+1}\|_M} \le \mathcal{O}(\sin\theta_{k+1})$, i.e.,$|\alpha(y^*_{k+1})| \le \mathcal{O}(\sin\theta_{k+1})$.

Let α^*_k be the minimizer of $\rho(x_k + \alpha p_k)$ and y^*_k be the global minimizer in U_k, which contains the all the vectors of the form $y = \beta_y F^m x_0 + \gamma_y g_y$. Note that $\rho(x_k + \alpha(y^*_{k+1})p_k) \ge \rho(x_k + \alpha^*_k p_k) = \rho(x_{k+1})$ and $\rho(y(y^*_{k+1})) \ge \rho(y^*_k)$.

It follows from (A.19) that

$$\begin{aligned}
\frac{1}{2}\rho(y^*_{k+1}) &= \frac{1}{2}\rho(x_k + \alpha(y^*_{k+1})p_k) + \frac{1}{2}(\rho(y(y^*_{k+1})) - \rho(x_k)) + R_3(\rho(y(y^*_{k+1})), \alpha(y^*_{k+1})p_k) \\
&\quad - R_3(\rho(x_k), \alpha(y^*_{k+1})p_k) + \mathcal{O}(\alpha(y^*_{k+1})\sin^2\theta_0) + \mathcal{O}(\alpha(y^*_{k+1})^2\sin^2\theta_0) \\
&\geq \frac{1}{2}\rho(x_k + \alpha^*_k p_k) + \frac{1}{2}(\rho(y(y^*_{k+1})) - \rho(x_k)) + R_3(\rho(y(y^*_{k+1})), \alpha(y^*_{k+1})p_k) \\
&\quad\quad - R_3(\rho(x_k), \alpha(y^*_{k+1})p_k) + \mathcal{O}(\alpha(y^*_{k+1})\sin^2\theta_0) + \mathcal{O}(\alpha(y^*_{k+1})^2\sin^2\theta_0) \\
&\geq \frac{1}{2}\rho(x_{k+1}) + \frac{1}{2}(\rho(y^*_k) - \rho(x_k)) + R_3(\rho(y(y^*_{k+1})), \alpha(y^*_{k+1})p_k) \\
&\quad\quad - R_3(\rho(x_k), \alpha(y^*_{k+1})p_k) + \mathcal{O}(\alpha(y^*_{k+1})\sin^2\theta_0) + \mathcal{O}(\alpha(y^*_{k+1})^2\sin^2\theta_0) \\
&\geq \frac{1}{2}\rho(x_{k+1}) + \frac{1}{2}(\rho(y^*_k) - \rho(x_k)) + \mathcal{O}(\sin\theta_{k+1})\mathcal{O}(\sin^2\theta_0).
\end{aligned}$$

Equivalently,

$$\rho(x_{k+1}) - \rho(y^*_{k+1}) \leq \rho(x_k) - \rho(y^*_k) + \mathcal{O}(\sin\theta_{k+1})\mathcal{O}(\sin^2\theta_0).$$

□

Bibliography

[1] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008.

[2] Sadhan K Adhikari. Numerical solution of the two-dimensional Gross–Pitaevskii equation for trapped interacting atoms. *Physics Letters A*, 265(1-2):91–96, 2000.

[3] Kumud Ajmani, Wing-Fai Ng, and Meng-Sing Liou. Preconditioned conjugate gradient methods for the Navier-Stokes equations. *Journal of Computational Physics*, 110(1):68–81, 1994.

[4] Patrick R Amestoy, Timothy A Davis, and Iain S Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905, 1996.

[5] Donald G Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM (JACM)*, 12(4):547–560, 1965.

[6] Donald GM Anderson. Comments on "Anderson acceleration, mixing and extrapolation". *Numerical Algorithms*, 80:135–234, 2019.

[7] Mike H Anderson, Jason R Ensher, Michael R Matthews, Carl E Wieman, and Eric A Cornell. Observation of Bose-Einstein condensation in a dilute atomic vapor. *Science*, 269(5221):198–201, 1995.

[8] Xavier Antoine, Weizhu Bao, and Christophe Besse. Computational methods for the dynamics of the nonlinear Schrödinger/Gross–Pitaevskii equations. *Computer Physics Communications*, 184(12):2621–2633, 2013.

[9] Xavier Antoine and Romain Duboscq. Robust and efficient preconditioned Krylov spectral solvers for computing the ground states of fast rotating and strongly interacting Bose–Einstein condensates. *Journal of Computational Physics*, 258:509–523, 2014.

[10] Xavier Antoine and Romain Duboscq. Modeling and computation of Bose-Einstein condensates: stationary states, nucleation, dynamics, stochasticity. *Nonlinear Optical and Atomic Systems: at the Interface of Physics and Mathematics*, pages 49–145, 2015.

[11] Xavier Antoine, Antoine Levitt, and Qinglin Tang. Efficient spectral computation of the stationary states of rotating Bose–Einstein condensates by preconditioned nonlinear conjugate gradient methods. *Journal of Computational Physics*, 343:92–109, 2017.

[12] Xavier Antoine, Qinglin Tang, and Yong Zhang. A preconditioned conjugated gradient method for computing ground states of rotating dipolar Bose-Einstein condensates via kernel truncation method for dipole-dipole interaction evaluation. *Communications in Computational Physics*, 24(4):966–988, 2018.

[13] Larry Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16(1):1–3, 1966.

[14] Zhaojun Bai and Ren-Cang Li. Minimization principles for the linear response eigenvalue problem I: Theory. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1075–1100, 2012.

[15] Zhaojun Bai and Ren-Cang Li. Minimization principles for the linear response eigenvalue problem II: Computation. *SIAM Journal on Matrix Analysis and Applications*, 34(2):392–416, 2013.

[16] Weizhu Bao and Yongyong Cai. Mathematical theory and numerical methods for Bose-Einstein condensation. *Kinetic and Related Models*, 6, 2013.

[17] Weizhu Bao and Qiang Du. Computing the ground state solution of Bose–Einstein condensates by a normalized gradient flow. *SIAM Journal on Scientific Computing*, 25(5):1674–1697, 2004.

[18] Weizhu Bao and Xinran Ruan. Computing ground states of Bose–Einstein condensates with higher order interaction via a regularized density function formulation. *SIAM Journal on Scientific Computing*, 41(6):B1284–B1309, 2019.

[19] Weizhu Bao and Weijun Tang. Ground-state solution of Bose–Einstein condensate by directly minimizing the energy functional. *Journal of Computational Physics*, 187(1):230–254, 2003.

[20] D Baye and J-M Sparenberg. Resolution of the Gross-Pitaevskii equation with the imaginary-time method on a lagrange mesh. *Physical Review E*, 82(5):056701, 2010.

[21] Peter Benner and Carolin Penke. Efficient and accurate algorithms for solving the Bethe-Salpeter eigenvalue problem for crystalline systems. *Journal of Computational and Applied Mathematics*, 400:113650, 2022.

[22] Gregory Beylkin, James M Keiser, and Lev Vozovoi. A new class of time discretization schemes for the solution of nonlinear PDEs. *Journal of Computational Physics*, 147(2):362–387, 1998.

[23] Jiri Brabec, Lin Lin, Meiyue Shao, Niranjan Govind, Chao Yang, Yousef Saad, and Esmond G Ng. Efficient algorithms for estimating the absorption spectrum within linear response TDDFT. *Journal of Chemical Theory and Computation*, 11(11):5197–5208, 2015.

[24] Cl C Bradley, CA Sackett, JJ Tollett, and Randall G Hulet. Evidence of Bose-Einstein condensation in an atomic gas with attractive interactions. *Physical Review Letters*, 75(9):1687, 1995.

[25] Tim Byrnes, Kai Wen, and Yoshihisa Yamamoto. Macroscopic quantum computation using Bose-Einstein condensates. *Physical Review A*, 85(4):040306, 2012.

[26] Marco Caliari, Alexander Ostermann, Stefan Rainer, and Mechthild Thalhammer. A minimisation approach for computing the ground state of Gross–Pitaevskii systems. *Journal of Computational Physics*, 228(2):349–360, 2009.

[27] Maria Mercede Cerimele, Maria Luisa Chiofalo, Francesca Pistella, Sauro Succi, and Mario P Tosi. Numerical solution of the Gross-Pitaevskii equation using an explicit finite-difference scheme: An application to trapped Bose-Einstein condensates. *Physical Review E*, 62(1):1382, 2000.

[28] Qing Cheng, Chun Liu, and Jie Shen. A new interface capturing method for Allen-Cahn type equations based on a flow dynamic approach in Lagrangian coordinates, I. One-dimensional case. *Journal of Computational Physics*, 419:109509, 2020.

[29] MARIA LUISA Chiofalo, Sauro Succi, and MP Tosi. Ground state of trapped interacting Bose-Einstein condensates by an explicit imaginary-time algorithm. *Physical Review E*, 62(5):7438, 2000.

[30] Nicolas Condette, Christof Melcher, and Endre Süli. Spectral approximation of pattern-forming nonlinear evolution equations with double-well potentials of quadratic growth. *Mathematics of Computation*, 80(273):205–223, 2011.

[31] Steven M Cox and Paul C Matthews. Exponential time differencing for stiff systems. *Journal of Computational Physics*, 176(2):430–455, 2002.

[32] Franco Dalfovo, Stefano Giorgini, Lev P Pitaevskii, and Sandro Stringari. Theory of Bose-Einstein condensation in trapped gases. *Reviews of Modern Physics*, 71(3):463, 1999.

[33] Ionut Danaila and Frédéric Hecht. A finite element method with mesh adaptivity for computing vortex states in fast-rotating Bose–Einstein condensates. *Journal of Computational Physics*, 229(19):6946–6960, 2010.

[34] Ionut Danaila and Parimah Kazemi. A new Sobolev gradient method for direct minimization of the Gross–Pitaevskii energy with rotation. *SIAM Journal on Scientific Computing*, 32(5):2447–2467, 2010.

[35] Kendall B Davis, M-O Mewes, Michael R Andrews, Nicolaas J van Druten, Dallin S Durfee, DM Kurn, and Wolfgang Ketterle. Bose-Einstein condensation in a gas of sodium atoms. *Physical Review Letters*, 75(22):3969, 1995.

[36] Claude M Dion and Eric Cancès. Ground state of the time-independent Gross–Pitaevskii equation. *Computer Physics Communications*, 177(10):787–798, 2007.

[37] Qiang Du, Lili Ju, Xiao Li, and Zhonghua Qiao. Maximum principle preserving exponential time differencing schemes for the nonlocal Allen–Cahn equation. *SIAM Journal on Numerical Analysis*, 57(2):875–898, 2019.

[38] Qiang Du and Roy A Nicolaides. Numerical analysis of a continuum model of phase transition. *SIAM Journal on Numerical Analysis*, 28(5):1310–1322, 1991.

[39] Jed A Duersch, Meiyue Shao, Chao Yang, and Ming Gu. A robust and efficient implementation of LOBPCG. *SIAM Journal on Scientific Computing*, 40(5):C655–C676, 2018.

[40] Stanley C Eisenstat and Ilse CF Ipsen. Three absolute perturbation bounds for matrix eigenvalues imply relative bounds. *SIAM Journal on Matrix Analysis and Applications*, 20(1):149–158, 1998.

[41] David J Eyre. Unconditionally gradient stable time marching the Cahn-Hilliard equation. *MRS Online Proceedings Library (OPL)*, 529:39, 1998.

[42] Haw-Ren Fang and Yousef Saad. A filtered Lanczos procedure for extreme and interior eigenvalue problems. *SIAM Journal on Scientific Computing*, 34(4):A2220–A2246, 2012.

[43] Xiaobing Feng and Andreas Prohl. Numerical analysis of the Allen-Cahn equation and approximation for mean curvature flows. *Numerische Mathematik*, 94:33–65, 2003.

[44] Xiaobing Feng and Andreas Prohl. Error analysis of a mixed finite element method for the Cahn-Hilliard equation. *Numerische Mathematik*, 99:47–84, 2004.

[45] Alexander L Fetter, B Jackson, and S Stringari. Rapid rotation of a Bose-Einstein condensate in a harmonic plus quartic trap. *Physical Review A*, 71(1):013605, 2005.

[46] Dale G Fried, Thomas C Killian, Lorenz Willmann, David Landhuis, Stephen C Moss, Daniel Kleppner, and Thomas J Greytak. Bose-Einstein condensation of atomic hydrogen. *Physical Review Letters*, 81(18):3811, 1998.

[47] Iulia Georgescu. 25 years of BEC. *Nature Reviews Physics*, 2(8):396–396, 2020.

[48] R Glowinski, B Mantel, J Periaux, G Poirier, and O Pironneau. An efficient preconditioned conjugate gradient method applied to nonlinear problems in fluid dynamics via least square formulations. *In: Computing Methods in Applied Sciences and Engineering.(A82-17551 06-34) Amsterdam*, pages 445–487, 1980.

[49] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

[50] Allan Griffin, David W Snoke, and Sandro Stringari. *Bose-Einstein condensation*. Cambridge University Press, 1996.

[51] Zhen-Chen Guo, Eric Chu, and Wen-Wei Lin. Doubling algorithm for the discretized Bethe-Salpeter eigenvalue problem. *Mathematics of Computation*, 88(319):2325–2350, 2019.

[52] William W Hager and Hongchao Zhang. A survey of nonlinear conjugate gradient methods. *Pacific journal of Optimization*, 2(1):35–58, 2006.

[53] Godfrey Harold Hardy, John Edensor Littlewood, and George Pólya. *Inequalities*. Cambridge university press, 1952.

[54] Harold V Henderson and Shayle R Searle. On deriving the inverse of a sum of matrices. *Siam Review*, 23(1):53–60, 1981.

[55] Magnus R Hestenes, Eduard Stiefel, et al. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

[56] Ulrich Hetmaniuk and Rich Lehoucq. Basis selection in LOBPCG. *Journal of Computational Physics*, 218(1):324–332, 2006.

[57] Marlis Hochbruck and Alexander Ostermann. Explicit exponential Runge–Kutta methods for semilinear parabolic problems. *SIAM Journal on Numerical Analysis*, 43(3):1069–1090, 2005.

[58] Elias Jarlebring, Simen Kvaal, and Wim Michiels. An inverse iteration method for eigenvalue problems with eigenvector nonlinearities. *SIAM Journal on Scientific Computing*, 36(4):A1978–A2001, 2014.

[59] B-W Jeng, Y-S Wang, and C-S Chien. A two-parameter continuation algorithm for vortex pinning in rotating Bose–Einstein condensates. *Computer Physics Communications*, 184(3):493–508, 2013.

[60] Lili Ju, Jian Zhang, and Qiang Du. Fast and accurate algorithms for simulating coarsening dynamics of Cahn–Hilliard equations. *Computational Materials Science*, 108:272–282, 2015.

[61] Jacek Kasprzak, Murielle Richard, S Kundermann, A Baas, P Jeambrun, Jonathan Mark James Keeling, FM Marchetti, MH Szymańska, R André, JL Staehli, et al. Bose–Einstein condensation of exciton polaritons. *Nature*, 443(7110):409–414, 2006.

[62] Daniel Kessler, Ricardo H Nochetto, and Alfred Schmidt. A posteriori error control for the Allen-Cahn problem: circumventing Gronwall's inequality. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 38(1):129–142, 2004.

[63] Andrew V Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM journal on Scientific Computing*, 23(2):517–541, 2001.

[64] AV Knyazev. A preconditioned conjugate gradient method for eigenvalue problems and its implementation in a subspace. In *Numerical Treatment of Eigenvalue Problems Vol. 5/Numerische Behandlung von Eigenwertaufgaben Band 5: Workshop in Oberwolfach, February 25–March 3, 1990/Tagung in Oberwolfach, 25. Februar–3. März 1990*, pages 143–154. Springer, 1991.

[65] Scott P Kolodziej, Mohsen Aznaveh, Matthew Bullock, Jarrett David, Timothy A Davis, Matthew Henderson, Yifan Hu, and Read Sandstrom. The suitesparse matrix collection website interface. *Journal of Open Source Software*, 4(35):1244, 2019.

[66] Sabine Korbel, Paul Boulanger, Ivan Duchemin, Xavier Blase, Miguel AL Marques, and Silvana Botti. Benchmark many-body GW and Bethe-Salpeter calculations for small transition metal molecules. *Journal of Chemical Theory and Computation*, 10(9):3934–3943, 2014.

[67] Georg Kresse and Jürgen Hafner. Ab initio molecular dynamics for liquid metals. *Physical Review B*, 47(1):558, 1993.

[68] Daniel Kressner, Marija Miloloža Pandur, and Meiyue Shao. An indefinite variant of LOBPCG for definite matrix pencils. *Numerical Algorithms*, 66(4):681–703, 2014.

[69] Ken Kreutz-Delgado. The complex gradient operator and the CR-calculus. *arXiv preprint arXiv:0906.4835*, 2009.

[70] Ruipeng Li, Yuanzhe Xi, Eugene Vecharynski, Chao Yang, and Yousef Saad. A thick-restart Lanczos algorithm with polynomial filtering for Hermitian eigenvalue problems. *SIAM Journal on Scientific Computing*, 38(4):A2512–A2534, 2016.

[71] Quan Liu, Ronald B Morgan, and Walter Wilcox. Polynomial preconditioned GMRES and GMRES-DR. *SIAM Journal on Scientific Computing*, 37(5):S407–S428, 2015.

[72] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[73] Giovanni Onida, Lucia Reining, and Angel Rubio. Electronic excitations: density-functional versus many-body Green's-function approaches. *Reviews of Modern Physics*, 74(2):601, 2002.

[74] Mike C Payne, Michael P Teter, Douglas C Allan, TA Arias, and ad JD Joannopoulos. Iterative minimization techniques for ab initio total-energy calculations: molecular dynamics and conjugate gradients. *Reviews of Modern Physics*, 64(4):1045, 1992.

[75] Carolin Penke, Andreas Marek, Christian Vorwerk, Claudia Draxl, and Peter Benner. High performance solution of skew-symmetric eigenvalue problems with applications in solving the Bethe-Salpeter eigenvalue problem. *Parallel Computing*, 96:102639, 2020.

[76] Elijah Polak and Gerard Ribiere. Note sur la convergence de méthodes de directions conjuguées. *Revue française d'informatique et de recherche opérationnelle. Série rouge*, 3(16):35–43, 1969.

[77] Sara Pollock and Leo G Rebholz. Anderson acceleration for contractive and noncontractive operators. *IMA Journal of Numerical Analysis*, 41(4):2841–2872, 2021.

[78] Lucia Reining, Valerio Olevano, Angel Rubio, and Giovanni Onida. Excitonic effects in solids described by time-dependent density-functional theory. *Physical Review Letters*, 88(6):066404, 2002.

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

[80] Thomas Schmelzer and L Trefethen. Evaluating matrix functions for exponential integrators via Carathéodory-Fejér approximation and contour integrals. *Electronic Transactions on Numerical Analysis*, 29:1–18, 2007.

[81] WG Schmidt, S Glutsch, PH Hahn, and F Bechstedt. Efficient $\mathcal{O}(N^2)$ method to solve the Bethe-Salpeter equation. *Physical Review B*, 67(8):085307, 2003.

[82] Meiyue Shao. Householder orthogonalization with a non-standard inner product. *arXiv preprint arXiv:2104.04180*, 2021.

[83] Meiyue Shao, H Felipe, Chao Yang, Jack Deslippe, and Steven G Louie. Structure preserving parallel algorithms for solving the Bethe–Salpeter eigenvalue problem. *Linear Algebra and its Applications*, 488:148–167, 2016.

[84] Jie Shen and Xiaofeng Yang. Numerical approximations of Allen-Cahn and Cahn-Hilliard equations. *Discrete Contin. Dyn. Syst*, 28(4):1669–1691, 2010.

[85] Jack Sherman and Winifred J Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950.

[86] Gerard LG Sleijpen and Henk A Van der Vorst. A Jacobi–Davidson iteration method for linear eigenvalue problems. *SIAM review*, 42(2):267–293, 2000.

[87] Gilbert W Stewart. *Matrix algorithms: volume 1: basic decompositions*. SIAM, 1998.

[88] John C Strikwerda. *Finite difference schemes and partial differential equations*. SIAM, 2004.

[89] Daniel Szyld and Fei Xue. Preconditioned eigensolvers for large-scale nonlinear Hermitian eigenproblems with variational characterizations. I. Extreme eigenvalues. *Mathematics of Computation*, 85(302):2887–2918, 2016.

[90] Zhongming Teng and Ren-Cang Li. Convergence analysis of Lanczos-type methods for the linear response eigenvalue problem. *Journal of Computational and Applied Mathematics*, 247:17–33, 2013.

[91] Zhongming Teng and Lei-Hong Zhang. A block Lanczos method for the linear response eigenvalue problem. *Electronic Transactions on Numerical Analysis*, 46:505–523, 2017.

[92] Zhongming Teng, Yunkai Zhou, and Ren-Cang Li. A block Chebyshev-Davidson method for linear response eigenvalue problems. *Advances in Computational Mathematics*, 42(5):1103–1128, 2016.

[93] Michael P Teter, Michael C Payne, and Douglas C Allan. Solution of Schrödinger's equation for large systems. *Physical Review B*, 40(18):12255, 1989.

[94] Tonghua Tian, Yongyong Cai, Xinming Wu, and Zaiwen Wen. Ground states of spin-F Bose–Einstein condensates. *SIAM Journal on Scientific Computing*, 42(4):B983–B1013, 2020.

[95] EV Tsiper. Variational procedure and generalized Lanczos recursion for small-amplitude classical oscillations. *Journal of Experimental and Theoretical Physics Letters*, 70(11):751–755, 1999.

[96] EV Tsiper. A classical mechanics technique for quantum linear response. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 34(12):L401, 2001.

[97] Henk A Van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2):631–644, 1992.

[98] Eugene Vecharynski, Jiri Brabec, Meiyue Shao, Niranjan Govind, and Chao Yang. Efficient block preconditioned eigensolvers for linear response time-dependent density functional theory. *Computer Physics Communications*, 221:42–52, 2017.

[99] Homer F Walker and Peng Ni. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735, 2011.

[100] Y-S Wang, B-W Jeng, and C-S Chien. A two-parameter continuation method for rotating two-component Bose-Einstein condensates in optical lattices. *Communications in Computational Physics*, 13(2):442–460, 2013.

[101] PES Wormer, F Visser, and J Paldus. Conjugate gradient method for the solution of linear equations: Application to molecular electronic structure calculations. *Journal of Computational Physics*, 48(1):23–44, 1982.

[102] Lingfei Wu, Fei Xue, and Andreas Stathopoulos. TRPL+K: Thick-restart preconditioned Lanczos+K method for large symmetric eigenvalue problems. *SIAM Journal on Scientific Computing*, 41(2):A1013–A1040, 2019.

[103] Xinming Wu, Zaiwen Wen, and Weizhu Bao. A regularized Newton method for computing ground states of Bose–Einstein condensates. *Journal of Scientific Computing*, 73:303–329, 2017.

[104] Fei Xue. One-step convergence of inexact Anderson acceleration for contractive and non-contractive mappings. *ETNA-Electronic Transactions on Numerical Analysis*, 55, 2021.

[105] Xingde Ye. The Legendre collocation method for the Cahn–Hilliard equation. *Journal of Computational and Applied Mathematics*, 150(1):87–108, 2003.

[106] Yongjun Yuan, Zhiguo Xu, Qinglin Tang, and Hanquan Wang. The numerical study of the ground states of spin-1 Bose-Einstein condensates with spin-orbit-coupling. *East Asian Journal on Applied Mathematics*, 8(3):598–610, 2018.

[107] Rong Zeng and Yanzhi Zhang. Efficiently computing vortex lattices in rapid rotating Bose–Einstein condensates. *Computer Physics Communications*, 180(6):854–860, 2009.

[108] Jian Zhang and Qiang Du. Numerical studies of discrete approximations to the Allen–Cahn equation in the sharp interface limit. *SIAM Journal on Scientific Computing*, 31(4):3042–3063, 2009.

[109] Hongxiu Zhong, Zhongming Teng, and Guoliang Chen. Weighted block Golub-Kahan-Lanczos algorithms for linear response eigenvalue problem. *Mathematics*, 7(1):53, 2019.

[110] Yunkai Zhou. A block Chebyshev–Davidson method with inner–outer restart for large eigenvalue problems. *Journal of Computational Physics*, 229(24):9188–9200, 2010.

[111] Yunkai Zhou and Yousef Saad. A Chebyshev-Davidson algorithm for large symmetric eigenproblems. *SIAM Journal on Matrix Analysis and Applications*, 29(3):954–971, 2007.