ABSTRACT

Title of dissertation: LINEAR STABILITY ANALYSIS USING LYAPUNOV INVERSE ITERATION

Minghao Wu, Doctor of Philosophy, 2012

Dissertation directed by: Professor Howard Elman
Department of Computer Science
Institute for Advanced Computer Studies

In this dissertation, we develop robust and efficient methods for linear stability analysis of large-scale dynamical systems, with emphasis on the incompressible Navier-Stokes equations. Linear stability analysis is a widely used approach for studying whether a steady state of a dynamical system is sensitive to small perturbations. The main mathematical tool that we consider in this dissertation is Lyapunov inverse iteration, a recently developed iterative method for computing the eigenvalue with smallest modulus of a special eigenvalue problem that can be specified in the form of a Lyapunov equation. It has the following "inner-outer" structure: the outer iteration is the eigenvalue computation and the inner iteration is solving a large-scale Lyapunov equation. This method has two applications in linear stability analysis: it can be used to estimate the critical value of a physical parameter at which the steady state becomes unstable (*i.e.*, sensitive to small perturbations), and it can also be applied to compute a few rightmost eigenvalues of the Jacobian matrix. We present numerical performance of Lyapunov inverse iteration

in both applications, analyze its convergence in the second application, and propose strategies of implementing it efficiently for each application.

In previous work, Lyapunov inverse iteration has been used to estimate the critical parameter value at which a parameterized path of steady states loses stability. We refine this method by proposing an adaptive stopping criterion for the Lyapunov solve (inner iteration) that depends on the accuracy of the eigenvalue computation (outer iteration). The use of such a criterion achieves dramatic savings in computational cost and does not affect the convergence of the target eigenvalue.

The method of previous work has the limitation that it can only be used at a stable point in the neighborhood of the critical point. We further show that Lyapunov inverse iteration can also be used to generate a few rightmost eigenvalues of the Jacobian matrix at any stable point. These eigenvalues are crucial in linear stability analysis, and existing approaches for computing them are not robust. A convergence analysis of this method leads to a way of implementing it that only entails one Lyapunov solve.

In addition, we explore the utility of various Lyapunov solvers in both applications of Lyapunov inverse iteration. We observe that different Lyapunov solvers should be used for the Lyapunov equations arising from the two applications. Applying a Lyapunov solver entails solving a number of large and sparse linear systems. We explore the use of sparse iterative methods for this task and construct a new variant of the Lyapunov solver that significantly reduces the costs of the sparse linear solves.

LINEAR STABILITY ANALYSIS USING LYAPUNOV INVERSE ITERATION

by

Minghao Wu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2012

Advisory Committee:
Professor Howard Elman, Chair/Advisor
Professor James Baeder, Dean's Representative
Professor David Levermore
Professor Ricardo Nochetto
Professor Dianne O'Leary

Acknowledgments

I am deeply grateful to my advisor, Howard Elman, for his patient guidance and constant encouragement. He has devoted a generous amount of time and energy to helping me with the difficulties in my research as well as to improving my writing. He has helped me believe that I was able to finish this dissertation.

I thank Karl Meerbergen and Alastair Spence for their work on Lyapunov inverse iteration, which is the starting point of my dissertation. The two visits Alastair paid to Maryland were extremely helpful to my research. I also thank him for enabling me to visit his department at The University of Bath for a very useful and enjoyable month.

I thank James Baeder, David Levermore, Ricardo Nochetto and Dianne O'Leary for serving on my committee and providing insightful comments and suggestions.

I thank Fei Xue for his willingness to help at all times, especially during the first two years of my graduate study.

Finally, I would like to thank my parents for their unconditional love and support and Sean for all the fun we had.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Stability analysis is the study of how steady-state solutions (or equilibria) of a dynamical system respond to small perturbations. Roughly speaking, if a steady-state solution is insensitive to all small perturbations, then it is considered to be stable; otherwise, it is unstable. Stability is a fundamental requirement for the practical use of a dynamical system because small perturbations exist almost everywhere in real life. For instance, products built by a factory will inevitably deviate from the original design made by an engineer. In real-world applications, instability can cause serious damage. One famous example is the Tacoma Narrows Bridge built in 1940, which oscillated considerably in windy conditions and finally collapsed only four months after its opening. Other examples include the flutter of the wings of an aircraft and the vibration of a skyscraper, both of which may lead to disasters.

For more than a century now, the standard mathematical approach of studying the stability of a steady-state solution for a dynamical system has been the following: given a steady-state solution to the dynamical system, perturb this solution by a small perturbation, linearize the system, and then check the rightmost eigenvalue (*i.e.*, the eigenvalue with algebraically largest real part) of the resulting linear operator. If the rightmost eigenvalue has negative real part, then the steady-state solution is stable; otherwise, it is unstable. This procedure is usually referred to as

linear stability analysis, and its success has been reported in many cases such as Rayleigh-Bénard convection [6] and Taylor-Couette flow [45].

In this thesis, a dynamical system refers to the system of ordinary differential equations (ODEs) arising from spatial discretization (finite difference, finite element, finite volume, *etc.*) of a system of two- or three-dimensional nonlinear partial differential equations (PDEs), such as the Navier-Stoke equations, for which there is considerable current interest in stability. We will assume that the discretization is good enough that it captures all the important features of the PDEs. Hence, stability of the continuous system can be inferred from the stability analysis for the discrete system.

Linear stability analysis of such a dynamical system eventually boils down to finding the rightmost eigenvalue of its Jacobian matrix. Direct methods such as the QR and QZ algorithms (see [44]) can be applied to compute the complete set of eigenpairs of a matrix. However, these methods are only suitable for matrices of order a few thousand at the most. As the dimension of the matrix grows, direct methods will quickly become forbidding. Our primary interest is the dynamical systems arising from spatial discretization of two- or three-dimensional PDEs, for which the order of the Jacobian matrix will be quite large. In the context of three-dimensional PDEs, for instance, the order of the Jacobian matrix typically varies from hundreds of thousands to millions.

Consequently, we have to turn to iterative eigenvalue solvers instead. Rather than finding all the eigenvalues of a matrix, iterative methods such as subspace iteration and Arnoldi's method (see [40, 44]) compute only a small subset of them. They

construct a small subspace by some means, project the matrix onto this subspace and compute using a direct method all the eigenpairs of the projection. From them, estimates of the eigenpairs of the original matrix can be obtained. In subspace iteration, by the way the subspace is constructed, it is rich in the eigenvectors associated with the dominant eigenvalues (*i.e.*, the eigenvalues with largest moduli), and these are eigenvalues that will be found. In Arnoldi's method, a Krylov subspace is built, and although the convergence properties of this method are less clear, it tends to converge to well-separated, extremal eigenvalues (see [38, 40]).

Unfortunately, when the dynamical system arises from spatial discretization of PDEs, the rightmost eigenvalue of the Jacobian matrix is in general neither dominant nor well-separated, and therefore it can be difficult to find if we simply apply an iterative method to the Jacobian matrix itself. Instead, the iterative method is usually applied to a transformation of the Jacobian matrix. The transformation should have the following qualities: the eigenvalues of the Jacobian matrix can be recovered easily from those of the transformed problem, and the rightmost eigenvalue of the Jacobian matrix will correspond to a well-separated and/or dominant eigenvalue of the transformed problem. Frequently used matrix transformations are shift-invert transformation, Cayley transformation and Chebyshev polynomials (see [28] for an overview).

Though they have been reported to be successful in linear stability analysis (see [28] and the references therein), a deficiency of matrix transformations is that their performance relies heavily on the choice of certain parameters and there is no good way of selecting them without *a priori* knowledge of the spectrum of the

Jacobian matrix. Poorly chosen parameters can result in convergence to a spurious eigenvalue, which in turn, may lead to misjudgement of the stability of the dynamical system. The shift-invert transformation, for instance, maps the eigenvalues of the Jacobian matrix closest to a complex number (called the 'shift') to the eigenvalues of the transformed problem with largest moduli. Thus, an ideal choice of the shift would be an estimate of the eigenvalue sought after, that is, the rightmost eigenvalue. However, such an estimate is usually not available in practice. This lack of robustness of existing iterative methods in the computation of the rightmost eigenvalue is essentially the motivation behind this thesis.

It is often the case that a dynamical system depends on a physical parameter, one well-known example being the Reynolds number in the Navier-Stokes equations. Consequently, the steady-state solution and hence the Jacobian matrix also depend on this parameter. A question of great interest to both mathematicians and engineers is how stability of the steady-state solution changes with this parameter; in particular, what the critical value of the parameter is at which the steady-state solution first becomes unstable. At this critical point, the rightmost eigenvalue of the Jacobian matrix is either zero or purely imaginary.

Besides the change in stability, another interesting phenomenon known as bifurcation also takes place at the critical point. A bifurcation point, loosely speaking, is where behavior of the solution changes qualitatively, and it is characterized by a zero eigenvalue or a conjugate pair of purely imaginary eigenvalues of the Jacobian matrix. At the critical point, it is the rightmost eigenvalue of the Jacobian matrix that is either zero or purely imaginary. If it is zero, with some additional

assumptions (see [19]), the critical point is a quadratic turning point, at which a branch of stable steady-state solutions 'turns around' and becomes unstable. If the dynamical system also possesses some kind of symmetry, then the critical point is a symmetry-breaking bifurcation point, where the existing branch of steady-state solutions becomes unstable and two new branches of steady-state solutions emerge that are symmetric with respect to the existing branch. If the Jacobian matrix has instead a conjugate pair of purely imaginary eigenvalues and some other requirements are also met (see [19]), the critical point is a Hopf bifurcation point, from which a periodic orbit originates. Quadratic turning point, symmetry-breaking bifurcation point and Hopf bifurcation point are three types of bifurcation points frequently seen in dynamical systems. Other types of bifurcation are discussed in [19].

There are various ways that one can calculate the critical point accurately when a good estimate of it is available (see [19] for a summary of these methods). However, getting such an estimate is a difficult task. The most commonly used approach is to monitor the rightmost eigenvalue of the Jacobian matrix while following a branch of stable steady-state solutions using numerical continuation (see [32]), until at some point the rightmost eigenvalue crosses the imaginary axis from the left to the right half of the complex plane. In this approach, we need to compute the rightmost eigenvalue for a series of Jacobian matrices corresponding to different values of the parameter.

As discussed earlier, choosing a transformation that favors the rightmost eigenvalue is difficult. If the critical point is a quadratic turning point or a symmetry-

breaking point, since we know that the rightmost eigenvalue at the critical point is zero, shift-invert transformation with a zero shift is a reliable choice. However, if the critical point is a Hopf bifurcation point, it is not so clear what transformation will be suitable as in general, we do not know where the rightmost eigenvalue may lie. It may have large imaginary part and thus could be far away from zero. In this case, if we continue to use shift-invert transformation with a zero shift, it is very likely that we will miss the rightmost eigenvalue. Because of this, the detection of the critical point of Hopf type is known to be a harder problem.

Due to the lack of robust methods for computing rightmost eigenvalues, an approach that is capable of detecting the loss of stability without the computation of rightmost eigenvalues is very much desirable. After all, what we are truly concerned with is whether the rightmost eigenvalue has crossed the imaginary axis or not, rather than its precise location. A method proposed in [20] falls into this category. Instead of finding the rightmost eigenvalue of the Jacobian matrix, this method monitors the so-called winding number (see [20]) of an analytic function along a branch of stable steady-state solutions. The poles of this function always coincide with the eigenvalues of the Jacobian matrix. As shown in [20], whenever a pole of this function crosses the imaginary axis from the left to the right half of the complex plane or a zero of it moves in the opposite way, its winding number will increase by π. The crossing of a pole (equivalently, an eigenvalue of the Jacobian matrix) and that of a zero can be distinguished by ways suggested in [20]. The advantage of the method of [20] is that it avoids any eigenvalue computation. However, in order to compute the winding number, many linear solves with complex coefficient matrices

are required, which can be extremely expensive for large dynamical systems. How to implement the ideas of [20] efficiently for such systems is still an open question.

Recently, another method aiming also at detecting the loss of stability without the computation of rightmost eigenvalues was proposed in [29]. Given a stable point in the vicinity of the critical point at which stability is lost, this method approximates the difference between the two parameter values, which in turn produces an estimate for the critical parameter value. It was shown in [29] that this distance is the eigenvalue with smallest modulus of a special eigenvalue problem that is in the form of a Lyapunov equation. To estimate the critical parameter value, it suffices to solve this eigenvalue problem for its eigenvalue with smallest modulus. Unlike for the rightmost eigenvalue, there are many robust methods for computing the eigenvalue with smallest modulus, in particular, inverse iteration (see [44]). The approach of applying inverse iteration to an eigenvalue problem with Lyapunov structure is referred to as Lyapunov inverse iteration. At each iteration of this method, we need to solve a large-scale Lyapunov equation with low-rank right-hand side, for which there exist many iterative solution methods such as the rational Krylov subspace method [12]. As by-products, this approach also produces estimates for the rightmost eigenvalue and its corresponding eigenvector of the Jacobian matrix at the critical point.

The ideas in [29] serve as the starting point of the work presented in this thesis. Basically, the following three questions are explored here.

(i) How can we improve the method of [29] so that it is more efficient when applied

to challenging problems?

(ii) Is it possible to develop a method that finds the rightmost eigenvalue at a stable point which may not be in the vicinity of the critical point?

(iii) How can we solve the linear systems arising from iterative solves of Lyapunov equations efficiently?

The first question concerns the practicality of the method proposed in [29]. The second one is motivated by the major limitation of the method of [29], that it only works at a stable point close to the critical point. The third question is important because the main cost of Lyapunov inverse iteration is solving a large-scale Lyapunov equation iteratively at each step, which in turn requires many large, sparse linear solves. The main results of this thesis are as follows.

With respect to the first question, we apply Lyapunov inverse iteration to identify Hopf bifurcations in models of incompressible flow including the driven-cavity flow [16], the Hopf point of which is notoriously difficult to find. In all the cases we consider, Lyapunov inverse iteration is able to locate the critical point successfully. These tests clearly indicate the robustness of the method proposed in [29]. In order to improve the efficiency of the method of [29], we build on an idea of [34], which shows that it is actually not necessary to solve the linear systems arising from inverse iteration accurately. Inspired by [34], we propose a new stopping criterion for solving the Lyapunov equations. When the refined algorithm is applied to the same examples, we observe that dramatic savings in computational cost can be achieved and more importantly, use of this stopping criterion does not affect

the convergence of the "outer iteration" for computing the target eigenvalue. This algorithm is also applied to several examples arising from aerodynamics [47], which further confirms its reliability and efficiency.

In answer to the second question, we develop a method for computing the distance between the rightmost eigenvalue and the imaginary axis at any stable point. This distance can be viewed as a qualitative indicator of how far away a stable point is from the critical point. We prove that it is the eigenvalue with smallest modulus of an eigenvalue problem similar in structure to the one considered in [29], so it can be computed using Lyapunov inverse iteration as well. An efficient way of implementing Lyapunov inverse iteration for this particular eigenvalue problem is also proposed, which reduces the total number of Lyapunov solves to one (*i.e.*, it guarantees that Lyapunov inverse iteration will converge in two steps). Furthermore, we show that a few rightmost eigenvalues can be obtained from Lyapunov inverse iteration for almost no additional cost.

As for the third question, we first note that solving a Lyapunov equation arising from Lyapunov inverse iteration iteratively entails solving a set of linear discrete PDEs. Thus, iterative solution methods developed for these problems can be applied directly to the linear systems arising from Lyapunov inverse iteration. We investigate the utility of iterative linear solvers [13, 14, 15] in Lyapunov inverse iteration applied to models of incompressible flows. Our numerical experiments show that solving the type of linear system arising from steady PDEs dominates the total cost of the iterative Lyapunov solve. Based on this observation, we modify the rational Krylov subspace method [12] in such a way that this type of linear solves can

mostly be avoided. The modification leads to significant savings in computational cost without degrading the convergence rate of the Lyapunov solver.

The plan of the remainder of this thesis is as follows. Chapter 2 gives a description of the type of problem we consider in this thesis and presents a review of iterative methods for computing the rightmost eigenvalue and Lyapunov inverse iteration. Chapter 3 presents the refined Lyapunov inverse iteration and describes its numerical performance for models of incompressible flow. Chapter 4 presents the new method for computing a few rightmost eigenvalues using Lyapunov inverse iteration for problems derived from parameters far from critical values, and describes its numerical performance for the benchmark problems considered above. Chapter 5 discusses how to solve the linear systems arising from Lyapunov solvers efficiently. Finally, Chapter 6 presents some concluding remarks.

Before concluding this introduction, we note that there are some limitations to linear stability analysis. In particular, there are examples for which the stability predicted by this method agrees poorly with that observed in the laboratory. One famous example is plane Couette flow, for which case linear stability analysis shows that the steady-state solution is always stable no matter how large the Reynolds number is [35], whereas instability can be observed for Reynolds numbers greater than certain threshold in the experiments [46]. For a long time, researchers have attributed this mismatch to the linearization step of linear stability analysis. Nevertheless, since the 1990s, the failure of linear stability analysis has been attributed to the nonorthogonality of the eigenvectors of the Jacobian matrix (see [48]). To be more specific, even when a steady-state solution is found to be stable by linear sta-

bility analysis, small perturbations can still grow to an arbitrarily large size before they decay. It is suggested in [48] that the rightmost eigenvalue of the pseudospectra of the Jacobian matrix is what truly dictates the stability of the steady-state solution, where pseudospectrum of a matrix is the collection of the eigenvalues of all the matrices whose 'distance' from this matrix is less than a small threshold.

Despite its failure in some cases, linear stability analysis has been successful in many other examples, and it is still a widely used method for studying stability of nonlinear dynamical systems. Moreover, although the numerical methods developed here are motivated by linear stability analysis, with some modifications, they are applicable to pseudospectral analysis as well. In particular, in [21], it is shown that analysis of pseudospectra can be performed by computing the rightmost eigenvalues of a sequence of problems similar in structure to those studied here. In this thesis, we focus on the classic problem of linear stability analysis.

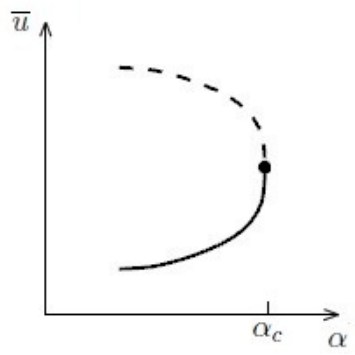
Chapter 2

Problem Statement and Survey of Existing Approaches

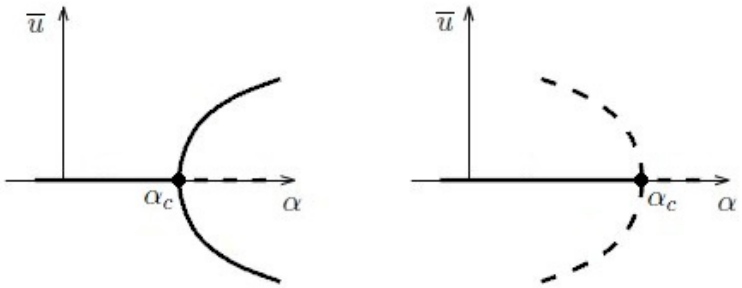
Consider the dynamical system

$$\mathbf{M}u_t = f(u, \alpha) \tag{2.1}$$

where $f : \mathbb{R}^n \times \mathbb{R} \mapsto \mathbb{R}^n$ is a nonlinear mapping, $u \in \mathbb{R}^n$ is the state variable (velocity, pressure, temperature, *etc.*), $\mathbf{M} \in \mathbb{R}^{n \times n}$ and α is a parameter. Such problems arise from spatial discretization of PDEs. The matrix $\mathbf{M}$ is usually called the mass matrix and could be singular. The dimension of the discretization, n, is usually large, especially for three-dimensional PDEs. Let $\overline{u}$ denote the steady-state solution to (2.1), *i.e.*, $\overline{u}_t = 0$. We are interested in the stability of $\overline{u}$: if a small perturbation $\delta(0)$ is introduced to $\overline{u}$ at time $t = 0$, does $\delta(t)$ grow with time (unstable), or does it decay (stable)? Let the solution path of the equilibrium equation $f(u, \alpha) = 0$ be the following set: $\mathcal{S} = \{(\overline{u}, \alpha) \,|f\,(\overline{u}, \alpha) = 0\}$. It is often the case that as the parameter α varies, there exists a critical point $\{(\overline{u}_c, \alpha_c)\} \in \mathcal{S}$ at which the steady-state solution $\overline{u}$ changes from being stable to unstable. Several examples of the critical points are displayed in Figure 2.1, in which the solid line represents a stable branch of $\mathcal{S}$ and the dashed line stands for an unstable branch. An important problem in applications is to find this critical parameter value α_c.



(a)



(b)

Figure 2.1: Illustration of the critical point

For any point $(\overline{u}, \alpha) \in \mathcal{S}$, linear stability of the steady-state solution $\overline{u}$ is determined by the spectrum of the generalized eigenvalue problem

$$\mathcal{J}x = \mu \mathbf{M}x \tag{2.2}$$

where $\mathcal{J} = \frac{\partial f}{\partial u}(\overline{u}, \alpha)$ is the Jacobian matrix of f evaluated at $(\overline{u}, \alpha)$. (If $\mathbf{M} = \mathbf{I}$, the identity matrix of order n, then (2.2) is a standard eigenvalue problem.) Typically both $\mathcal{J}$ and $\mathbf{M}$ are large and sparse, and $\mathcal{J}$ is in general nonsymmetric. If all the eigenvalues of (2.2) have strictly negative real part, then $\overline{u}$ is a stable steady solution; if some eigenvalues of (2.2) have nonnegative real part, then $\overline{u}$ is unstable. Therefore, a change of stability can be detected by monitoring the rightmost eigenvalue (*i.e.*,

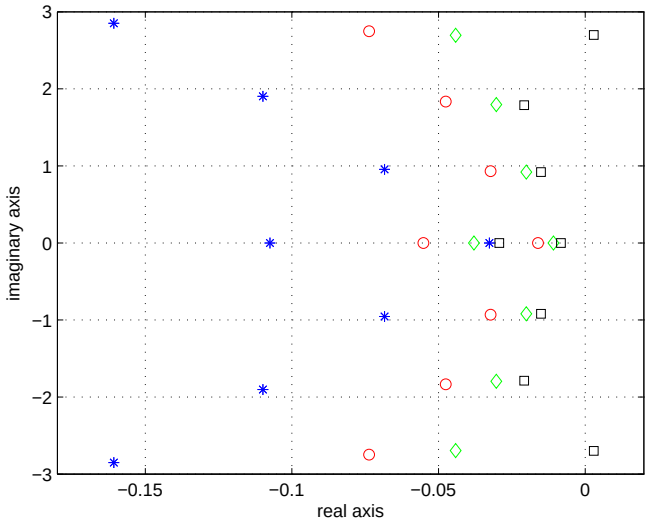


Figure 2.2: The evolution of a few rightmost eigenvalues of (2.2) for the parameter values $\alpha_1(*) < \alpha_2(\circ) < \alpha_3(\diamondsuit) < \alpha_4(\square)$

the eigenvalue with algebraically largest real part) of (2.2) while following a stable branch of $\mathcal{S}$ using numerical continuation (see [19, 32]). In practice, we usually keep track of the evolution of a few rightmost eigenvalues of (2.2) along this branch. What can typically be observed for the spectrum of (2.2) is that either all the eigenvalues of (2.2) lie in the left half of the complex plane, or a few of them have crossed the imaginary axis and have small positive real parts. We demonstrate in Figure 2.2 the evolution of the eight rightmost eigenvalues of (2.2) arising from the driven-cavity flow with α being the Reynolds number. (We defer a detailed discussion of this model to the next chapter.) It clearly indicates that the critical parameter value α_c lies between α_3 and α_4.

The critical point is also a bifurcation point, at which the behavior of $\mathcal{S}$ changes qualitatively. A bifurcation point is characterized by the existence of a zero eigenvalue or a conjugate pair of purely imaginary eigenvalues of (2.2). At the critical

point, the rightmost eigenvalue is either zero or purely imaginary. If it is zero and also simple (*i.e.*, both its algebraic and geometric multiplicity equal one), then under some generic conditions (see [19]), the critical point is a quadratic turning point. Figure 2.1(a) illustrates an example of such a bifurcation point, at which the stable solution branch 'turns around' and loses its stability. In addition, if symmetry is also present in the dynamical system (2.1), then the critical point is a symmetry-breaking bifurcation point. In Figure 2.1(b), we display two examples of this bifurcation point, near which the solution paths behave quite differently. (They are called a supercritical symmetry-breaking bifurcation point (left) and a subcritical symmetry-breaking bifurcation point (right), respectively.) For both examples shown in Figure 2.1(b), the symmetry is with respect to the α-axis. In either case, at the critical point, the stable solution branch that is symmetric loses its stability and meanwhile, two asymmetric branches originate that are symmetric with each other. On the other hand, if (2.2) has a conjugate pair of purely imaginary rightmost eigenvalues at the critical point that are simple, then given some additional assumptions (see also [19]), it is a Hopf bifurcation point. At this point, a periodic orbit of (2.1) arises and the existing branch of steady-state solution changes from being stable to unstable. Described above are three types of bifurcation that are commonly observed in the solution path of (2.1). More complicated types of bifurcation can be found in [19].

Remark. After the rightmost eigenvalue of (2.2) has crossed the imaginary axis and entered the right half of the complex plane, other eigenvalues may do the same as we keep following the unstable solution branch; in addition, the eigenvalues

that are already in the right half of the complex plane may cross the imaginary axis again and return to the left half of the complex plane. Consequently, multiple bifurcation points may exist along this branch. In this thesis, our focus is on the detection of the very first bifurcation point, namely, the critical point at which stability of the steady-state solution is lost.

In the following sections, we summarize two approaches for the detection of the critical point: computing the rightmost eigenvalue of (2.2) and Lyapunov inverse iteration. The former is the most commonly used method for linear stability analysis, and the latter is a new and more reliable approach on which the rest of this thesis is based.

2.1 The Computation of the Rightmost Eigenvalue

The critical point $(\overline{u}_c, \alpha_c)$ at which stability of the steady-state solution $\overline{u}$ of (2.1) is lost can be detected by monitoring the rightmost eigenvalue of (2.2) along a branch of the solution path $\mathcal{S}$. In this approach, we need to compute the rightmost eigenvalue of (2.2) for a set of sample points $(\overline{u}, \alpha)$ on this branch. Direct methods for (2.2) such as the QR algorithm and the QZ algorithm (see [44]) produce the complete set of eigenpairs of (2.2), and therefore, the rightmost eigenvalue of (2.2) is guaranteed to be found by them. However, these methods are only applicable for small eigenvalue problems, the dimension n being a few thousand at the most. Since our primary interest is linear stability analysis for the dynamical system (2.1) arising from spatial discretization of two- or three-dimensional PDEs, n is typically quite large and

direct methods are simply not feasible. Iterative eigenvalue solvers such as subspace iteration and Arnoldi's method ([40]) are efficient alternatives when only a small subset of the eigenvalues of (2.2) are wanted. When a few rightmost eigenvalues are sought, an iterative eigenvalue solver is usually applied to a transformed version of (2.2) instead to accelerate convergence. The choice of this transformation is crucial for the efficiency and reliability of the eigenvalue computation. In this section, we discuss both aspects of the computation of the rightmost eigenvalue: iterative eigenvalue solvers and matrix transformations.

2.1.1 Iterative Eigenvalue Solvers

Prior to our discussion on iterative eigenvalue solvers, we note that they cannot be applied directly to a generalized eigenvalue problem (2.2) with $\mathbf{M} \neq \mathbf{I}$. Therefore, we rewrite (2.2) into a standard eigenvalue problem first. Assume for now that the mass matrix $\mathbf{M}$ is nonsingular. Then (2.2) and the standard eigenvalue problem

$$Cx = \mu x \tag{2.3}$$

where $C = \mathbf{M}^{-1}\mathcal{J}$ have the same set of eigenpairs. In the case of models of incompressible flows, where $\mathbf{M}$ is singular, it is possible to modify $\mathbf{M}$ to produce a nonsingular variant; a discussion is again postponed to Chapter 3.

We consider two frequently used iterative eigenvalue solvers: subspace iteration and Arnoldi's method. Both method construct a small subspace of $\mathbb{R}^n$ and compute the complete set of eigenpairs of the orthogonal projection of C onto this subspace,

from which approximations to a small subset of the eigenpairs of C can be obtained. Since the subspace is small, the eigenpairs of the projection can be computed by the QR algorithm. In the following discussion, let $\{(\mu_j, x_j)\}_{j=1}^{n}$ denote the eigenpairs of C and assume that m ($\ll n$) eigenvalues of C are wanted.

Subspace iteration is equivalent to applying the power method [44] to a block of starting vectors simultaneously. The subspace built in this method is the one spanned by the columns of (orthonormalized) $C^{\ell-1}V_0$, where $V_0 = [v_1, v_2, \ldots, v_m] \in \mathbb{R}^{n\times m}$ is an orthonormal matrix that contains m starting vectors and ℓ is the number of steps of subspace iteration performed. Assume that $|\mu_1| \geq |\mu_2| \geq \cdots \geq |\mu_m| > |\mu_{m+1}| \geq \cdots \geq |\mu_n|$ and that the starting block V_0 is not deficient in the eigenspace of C associated with the m dominant eigenvalues (*i.e.*, eigenvalues with largest moduli). It is not difficult to see that as ℓ increases, subspace iteration will converge to $\{\mu_j\}_{j=1}^{m}$. This method is outlined in Algorithm 1.

Algorithm 1: Subspace iteration

1. Let $V_0 = [v_1, v_2, \ldots, v_m] \in \mathbb{R}^{n\times m}$ be an orthonormal matrix.
2. For $\ell = 1, 2, \cdots$
   - 2.1. $V_\ell \leftarrow CV_{\ell-1}$.
   - 2.2. Orthonormalize V_ℓ.
   - 2.3. Form $H_m = V_\ell^T C V_\ell$.
   - 2.4. Compute the eigenpairs (μ_i, y_i) $(i = 1, 2, \cdots, m)$ of H_m using the QR algorithm.
   - 2.5. Compute the approximate eigenpairs of C: $(\mu_i, V_\ell y_i)$ $(i = 1, 2, \cdots, m)$

In each iteration of Algorithm 1, m matrix-vector products with C are needed in order to compute $CV_{\ell-1}$. If the m dominant eigenvalues of C are well-separated from the rest of the eigenvalues, then the convergence of this method is rapid [40, 44].

Arnoldi's method constructs the Krylov subspace

$$\mathcal{K}_m(C, v_1) = span\left\{v_1, Cv_1, C^2v_1, \cdots, C^{m-1}v_1\right\} \tag{2.4}$$

where v_1 is a starting vector of unit norm. Similar to subspace iteration, this method generates m approximate eigenpairs of C by computing the eigenpairs of its projection onto (2.4). An implementation of this method is given by Algorithm 2.

Algorithm 2: Arnoldi's method

1. Let $v_1 \in \mathbb{R}^n$ with $\|v_1\|_2 = 1$.
2. For $\ell = 1, 2, \cdots, m$
   - 2.1. $w = Cv_\ell$.
     for $i = 1, \ldots, \ell$
     $h_{i,\ell} \leftarrow v_i^T w$;
     $w \leftarrow w - v_i h_{i,\ell}$.
   - 2.2. $h_{\ell+1,\ell} \leftarrow \|w\|_2$ and $v_{\ell+1} \leftarrow w/h_{\ell+1,\ell}$.
   - 2.3. Compute the eigenpairs (μ_i, y_i) $(i = 1, 2, \cdots, \ell)$ of H_ℓ using the QR algorithm, where $H_\ell = [h_{ij}]_{i,j=1}^{\ell}$.
   - 2.4. Compute the eigenpairs of C: $(\mu_i, V_\ell y_i)$ $(i = 1, 2, \cdots, \ell)$, where $V_\ell = [v_1, v_2, \cdots, v_\ell]$.

This method computes the *Arnoldi decomposition*

$$CV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T, \tag{2.5}$$

where the columns of V_m and v_{m+1} form an orthonormal basis of the Krylov subspace $\mathcal{K}_{m+1}(C, v_1)$, $H_m \in \mathbb{R}^{m\times m}$ is upper Hessenberg, and e_m is the last column of the identity matrix of order m. In subspace iteration, the $m \times m$ projection H_m of the large matrix C needs to be computed explicitly, whereas in Arnoldi's method, this matrix is formed in a clever way. One matrix-vector product with C is needed at

each iteration of Algorithm 2. Arnoldi's method often converges much more rapidly than subspace iteration, although it is not so clear which eigenvalues of C it will converge to. An analysis in [40] suggests that it tends to find the well-separated, extremal eigenvalues of C.

Rather than one run of Algorithm 2, Arnoldi's method is often restarted using converged eigenvectors (see [40, 44]). The most successful restarting strategy is the Implicitly Restarted Arnoldi's Method (IRA) ([43, 44]), in which the unwanted eigendirections (specified by the user) are filtered out from the Krylov subspace in a clever way. The techniques developed in Chapters 3 and 4 will be compared with this state-of-the-art eigenvalue solver.

2.1.2 Matrix Transformations

Iterative eigenvalue solvers such as subspace iteration and Arnoldi's method can be applied to compute a few well-separated, dominant eigenvalues of a matrix. Unfortunately, when the matrices $\mathcal{J}$ and $\mathbf{M}$ come from spatial discretization of PDEs, the rightmost eigenvalue of (2.2) is in general neither well-separated nor dominant. In order to accelerate the convergence of the rightmost eigenvalue, an iterative eigenvalue solver is often applied to a transformed problem

$$\mathcal{T}(\mathcal{J}, \mathbf{M})x = \theta x. \tag{2.6}$$

The matrix transformation $\mathcal{T}$ should be chosen such that the rightmost eigenvalue of (2.2) is mapped to a well-separated and dominant eigenvalue of (2.6) and the

eigenpairs of (2.2) can be recovered from those of (2.6) easily. In addition, the matrix-vector product with $\mathcal{T}(\mathcal{J}, \mathbf{M})$ should be easy to compute. Several types of matrix transformation can be used to accelerate the convergence of the rightmost eigenvalue, such as shift-invert transformation [40], Cayley transformation [17] and Chebyshev polynomial [40]. An overview of these methods is given in [28]. In this review, we consider shift-invert transformation and Cayley transformation. Let (μ_j, x_j) $(j = 1, 2, \ldots, n)$ denote the eigenpairs of (2.2) with $Re\,(\mu_1) \geq Re\,(\mu_2) \geq \cdots \geq Re\,(\mu_n)$. Then μ_1 is the rightmost eigenvalue.

Shift-invert transformation is defined to be

$$\mathcal{T}(\mathcal{J}, \mathbf{M}) = (\mathcal{J} - s\mathbf{M})^{-1}\mathbf{M} \tag{2.7}$$

where $s \in \mathbb{C}$ is called a shift. The eigenpairs of (2.7) are (θ_j, x_j) $(j = 1, 2, \cdots, n)$, where θ_j is given by

$$\theta_j = \frac{1}{\mu_j - s}. \tag{2.8}$$

This transformation enhances the eigenvalues of (2.2) near the shift s (see Figure 2.3). If we apply subspace iteration or Arnoldi's method to (2.7), then the eigenvalues of (2.6) that correspond to those eigenvalues are more likely to be found. When applied to (2.7), subspace iteration requires m solves with $\mathcal{J} - s\mathbf{M}$ each step, and Anorldi's method requires one solve with $\mathcal{J} - s\mathbf{M}$ *per* iteration.

Since we are interested in finding the rightmost eigenvalue μ_1 of (2.2), an ideal choice for the shift s would be its estimate. However, such an estimate is in general

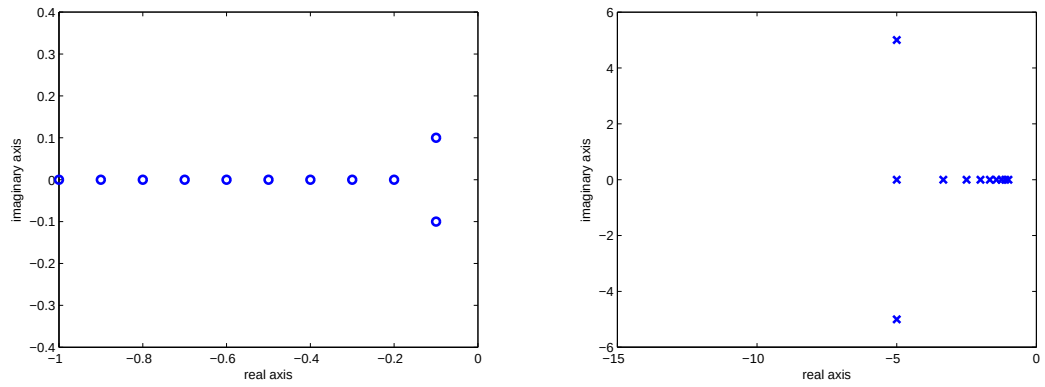


Figure 2.3: Shift-invert transformation. Left: the eigenvalues of (2.2) (denoted by $\circ$). Right: the eigenvalues of (2.7) with $s = 0$ (denoted by $\times$).

not available. When no *a priori* information about the location of μ_1 is available, s is often chosen to be zero. Shift-invert transformation with a zero shift maps the eigenvalues of (2.2) away from zero to the eigenvalues of (2.7) with small moduli and the eigenvalues of (2.2) close to zero to the dominant eigenvalues of (2.7). The benefits of this strategy are two fold. When the matrices $\mathcal{J}$ and $\mathbf{M}$ arise from spatial discretization of PDEs, (2.2) typically has a large number of eigenvalues in the left half of the complex plane that are away from the imaginary axis and thus irrelevant in linear stability analysis. They will be mapped to the $\theta_j{}'s$ clustered at zero, which will not be computed by an iterative eigenvalue solver. Furthermore, if μ_1 is real or has small imaginary part, it will be mapped to a dominant eigenvalue of (2.3) that can be found by an iterative method easily (see Figure 2.3 for an example).

If the critical point is a quadratic turning point or a symmetry-breaking bifurcation point, then zero is an effective choice for the shift in the shift-invert transformation (2.7). However, in the case of Hopf bifurcation, μ_1 may have imaginary part so large that it is further away from zero than many other eigenvalues close to the

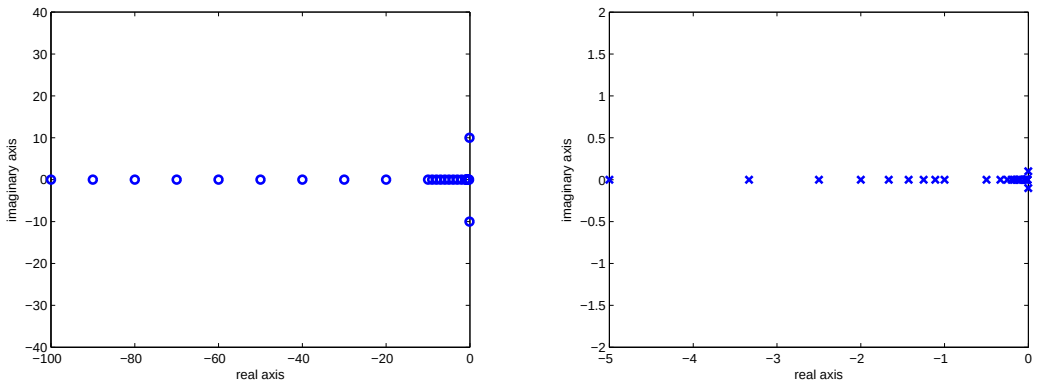


Figure 2.4: Shift-invert transformation when the rightmost eigenvalue has a large imaginary part. Left: the eigenvalues of (2.2) (denoted by $\circ$). Right: the eigenvalues of (2.7) with $s = 0$ (denoted by $\times$).

imaginary axis (see Figure 2.4). Shift-invert transformation with a zero shift will map these eigenvalues to the eigenvalues of (2.6) with moduli larger than $|\theta_1|$. Using a large subspace in the iterative eigenvalue solver increases the chances of finding θ_1. However, a large subspace implies a large number of linear solves with $\mathcal{J}$ and more importantly, it is difficult to determine how large the subspace should be to guarantee that θ_1 will be computed.

Shift-invert transformation emphasizes the eigenvalues near a given point in the complex plane. Cayley transformation, on the other hand, enhances the eigenvalues to the right of a given line in the complex plane. Therefore, it is more suitable for the computation of the rightmost eigenvalues. The definition of Cayley transformation is

$$\mathcal{T}(\mathcal{J}, \mathbf{M}) = (\mathcal{J} - s\mathbf{M})^{-1}(\mathcal{J} - t\mathbf{M}) \tag{2.9}$$

with $s, t \in \mathbb{C}$. The eigenpairs of (2.9) are (θ_j, x_j), $(j = 1, 2, \cdots, n)$ where

$$\theta_j = \frac{\mu_j - t}{\mu_j - s}. \tag{2.10}$$

A nice property of Cayley transformation is the following:

$$\begin{aligned} &\text{if } Re\,(\mu_j) < \frac{1}{2}(s+t), \text{ then } |\theta_j| < 1; \\ &\text{if } Re\,(\mu_j) = \frac{1}{2}(s+t), \text{ then } |\theta_j| = 1, \text{ and} \\ &\text{if } Re\,(\mu_j) > \frac{1}{2}(s+t), \text{ then } |\theta_j| > 1. \end{aligned} \tag{2.11}$$

As in the case of shift-invert transformation, if m eigenvalues of (2.9) are wanted, subspace iteration requires m solves with $\mathcal{J} - s\mathbf{M}$ each step, and Arnoldi's method requires one solve with $\mathcal{J} - s\mathbf{M}$ each step. Note that

$$(\mathcal{J} - s\mathbf{M})^{-1}(\mathcal{J} - t\mathbf{M}) = \mathbf{I} + (s-t)(\mathcal{J} - s\mathbf{M})^{-1}\mathbf{M}.$$

That is, Cayley transformation is scaled and translated shift-invert transformation. Since Arnoldi's method is translation invariant [30], applying it to (2.7) and (2.9) generates the exact same subspace in exact arithmetic. In other words, Anorldi's method is insensitive to the choice of t.

Suppose for now that we know the complete set of eigenvalues of (2.2), namely, $\{\mu_j\}_{j=1}^n$. If we want to compute the m rightmost eigenvalues of (2.2), an ideal choice for s and t is such that

$$\frac{1}{2}(s+t) = Re\,(\mu_{m+1})\,. \tag{2.12}$$

This way, by (2.11), the m rightmost eigenvalues of (2.2) will be mapped outside of the unit circle and the rest of the eigenvalues of (2.2) will be mapped inside of the unit circle. Consequently, $\{\theta_j\}_{j=1}^m$ are guaranteed to be found by subspace iteration. Another equation besides (2.12) is needed to completely determine s and t. For instance, in [17], s and t are selected such that $|\theta_1|$ is maximized.

In practice, however, the choice for s and t is not so straightforward since $\{\mu_j\}_{j=1}^n$ are not known. A preprocessing step is often needed in order to obtain approximate spectral information of (2.2) (see [17, 28]). For example, a few steps of subspace iteration can be applied to $\mathcal{J}^{-1}\mathbf{M}$ (*i.e.*, shift-invert transformation with a zero shift) to obtain some eigenvalue estimates of (2.2) (see [28]). The shifts s and t are then chosen based on such information. As pointed out in the discussion of shift-invert transformation, applying an iterative eigenvalue solver to $\mathcal{J}^{-1}\mathbf{M}$ has the risk of emphasizing the wrong eigenvalues. This leads to a poor choice of s and t that causes θ_1 to be missed by an iterative eigenvalue solver.

We end the discussion on matrix transformations with the following remarks.

1. When a proper matrix transformation is used, iterative eigenvalue solvers, especially Arnoldi's method and its variants, are very efficient in the computation of the rightmost eigenvalue of (2.2).

2. When the rightmost eigenvalue of (2.2) is not real, it can be rather difficult to find, especially when its imaginary part is large. This is due to the lack of estimates for the rightmost eigenvalue, without which an effective matrix transformation cannot be constructed.

2.2 Lyapunov Inverse Iteration

As seen in the previous section, there is no robust way of computing the rightmost eigenvalue of (2.2). A method of detecting the critical point that avoids such computation is wanted. It is shown in [29] that the critical parameter value α_c can be estimated by solving an eigenvalue problem in the form of a Lyapunov equation for its eigenvalue with smallest modulus. A version of inverse iteration [44] referred to as Lyapunov inverse iteration is proposed in [29] for computing this eigenvalue. As by-products of Lyapunov inverse iteration, we can also obtain estimates for the rightmost eigenvalue of (2.2) at the critical point and the eigenvector associated with it. We will focus on the case in which the stability of the steady-state solution is lost to a Hopf bifurcation point, although the ideas in [29] are also applicable to the case where instability is caused by a real eigenvalue crossing the imaginary axis.

2.2.1 An Eigenvalue Problem with Lyapunov Structure

Assume that $(\overline{u}_0, \alpha_0)$ is a point on a stable branch of the solution path $\mathcal{S}$, and that the Hopf point $(\overline{u}_c, \alpha_c)$ at which stability is lost lies in its neighborhood. As can be seen from Figure 2.1, one value of the parameter α may correspond to multiple steady states $\overline{u}$. Therefore, $\overline{u}$ is in general not a function of α. However, if restricted to a single branch of $\mathcal{S}$, $\overline{u}$ is indeed a function of α, *i.e.*, $\overline{u} = \overline{u}(\alpha)$. As a result, on the stable branch that contains $(\overline{u}_0, \alpha_0)$, the Jacobian matrix can be written as

$$\mathcal{J}(\overline{u}, \alpha) = \mathcal{J}(\overline{u}(\alpha), \alpha) = \mathcal{J}(\alpha).$$

At any point on this branch and in the neighborhood of $(\overline{u}_0, \alpha_0)$, $\mathcal{J}(\alpha)$ can be approximated as

$$\mathcal{J}(\alpha_0) + (\alpha - \alpha_0)\frac{d\mathcal{J}}{d\alpha}(\alpha_0) = \mathbf{A} + \lambda_\alpha \mathbf{B},$$

where $\mathbf{A}$, $\mathbf{B}$ are known and λ_α is an unknown quantity that characterizes the distance from $(\overline{u}_0, \alpha_0)$ to $(\overline{u}, \alpha)$. In particular, the Jacobian matrix at the Hopf point can be approximated by $\mathbf{A} + \lambda_c \mathbf{B}$, where $\lambda_c = \alpha_c - \alpha_0$. The critical value α_c can then be approximated by computing λ_c.

We assume for simplicity that $\mathcal{J}(\alpha) = \mathbf{A} + \lambda_\alpha \mathbf{B}$ in the neighborhood of (u_0, α_0). Consider the parameterized eigenvalue problem

$$(\mathbf{A} + \lambda \mathbf{B})x = \mu \mathbf{M} x \tag{2.13}$$

When $\lambda = \lambda_c$, (2.13) has a conjugate pair of purely imaginary rightmost eigenvalues $(\beta i, -\beta i)$ with $\beta > 0$. Since the critical point $(\overline{u}_c, \alpha_c)$ is the bifurcation point closest to the stable point $(\overline{u}_0, \alpha_0)$, λ_c is the value of λ closest to zero such that (2.13) has a pair of eigenvalues that sum to zero.

The following theorem is the main theoretical motivation for the techniques in [29]:

Theorem 2.1. *Assume* $\mathbf{M}$ *is nonsingular, and* μ_1, μ_2 $(\mu_1 \neq \mu_2)$ *are simple eigenvalues of (2.2) whose corresponding eigenvectors are* x_1, x_2*. The following two statements are equivalent:*

1. *zero is a double eigenvalue of* $\mathcal{J} \otimes \mathbf{M} + \mathbf{M} \otimes \mathcal{J}$ *that corresponds to the eigen-*

vector $\xi_1 x_1 \otimes x_2 + \xi_2 x_2 \otimes x_1$*, for any* $\xi_1, \xi_2 \in \mathbb{C}$*;*

2. (μ_1, μ_2) *is the only pair of eigenvalues of (2.2) that sums to zero.*

Proof. Since $\mathbf{M}$ is nonsingular, by properties of Kronecker product, $\mathbf{M} \otimes \mathbf{M}$ is nonsingular. Let $\{(\mu_j, x_j)\}_{j=1}^n$ be the eigenpairs of (2.2) and let J be the Jordan normal form of $\mathbf{M}^{-1}\mathcal{J}$. By properties of Kronecker product,

$$(\mathbf{M} \otimes \mathbf{M})^{-1}(\mathcal{J} \otimes \mathbf{M} + \mathbf{M} \otimes \mathcal{J}) \tag{2.14}$$

and $J \otimes I + I \otimes J$ are similar, where I is the identity matrix of order n. Thus, the eigenvalues of (2.14) are $\{\mu_i + \mu_j\}_{i,j=1}^n$ $(i, j = 1, 2, \ldots, n)$. For any pair (i, j),

$$(\mathbf{M} \otimes \mathbf{M})^{-1}(\mathcal{J} \otimes \mathbf{M} + \mathbf{M} \otimes \mathcal{J})(x_i \otimes x_j) = \mathbf{M}^{-1}\mathcal{J}x_i \otimes x_j + x_i \otimes \mathbf{M}^{-1}\mathcal{J}x_j$$

$$= \mu_i x_i \otimes x_j + x_i \otimes \mu_j x_j = (\mu_i + \mu_j)(x_i \otimes x_j).$$

Therefore, $x_i \otimes x_j$ is an eigenvector of (2.14) associated with the eigenvalue $\mu_i + \mu_j$. Similarly, we can show that $x_j \otimes x_i$ is also an eigenvector of (2.14) associated with $\mu_i + \mu_j$.

We first prove statement 2 given statement 1. Note that $\mathcal{J} \otimes \mathbf{M} + \mathbf{M} \otimes \mathcal{J}$ and (2.14) have the same null space. Thus, if statement 1 is true, then zero is also a double eigenvalue of (2.14) that corresponds to the eigenvector $\xi_1 x_1 \otimes x_2 + \xi_2 x_2 \otimes x_1$. Since the eigenvalues of (2.14) are $\{\mu_i + \mu_j\}_{i,j=1}^n$ and μ_1, μ_2 $(\mu_1 \neq \mu_2)$ are simple, (μ_1, μ_2) is the only pair of eigenvalues of (2.2) that sums to zero.

Now assume statement 2 is true. Since (μ_1, μ_2) $(\mu_1 \neq \mu_2)$ is the only pair of

eigenvalues of (2.2) that sums to zero and both μ_1, μ_2 are simple, zero is a double eigenvalue of (2.14) with the eigenvector $\xi_1 x_1 \otimes x_2 + \xi_2 x_2 \otimes x_1$. Since $\mathcal{J} \otimes \mathbf{M} + \mathbf{M} \otimes \mathcal{J}$ and (2.14) have the same null space, statement 1 follows immediately. □

We continue to assume that $\mathbf{M}$ is nonsingular. In addition, assume that when $\lambda = \lambda_c$, the rightmost eigenvalues of (2.13) (βi and $-\beta i$) are simple and there are no other eigenvalues lying on the imaginary axis. Let v and $\overline{v}$ be the eigenvectors corresponding to βi and $-\beta i$. Since λ_c is the parameter closest to zero such that (2.13) has a pair of eigenvalues summing to zero, according to Theorem 2.1, λ_c is the parameter closest to zero such that $(\mathbf{A} + \lambda \mathbf{B}) \otimes \mathbf{M} + \mathbf{M} \otimes (\mathbf{A} + \lambda \mathbf{B})$ has a zero eigenvalue. Alternatively, λ_c is the eigenvalue closest to zero for the $n^2 \times n^2$ generalized eigenvalue problem

$$(\Delta_1 + \lambda \Delta_0) z = 0 \tag{2.15}$$

where

$$\Delta_1 = \mathbf{A} \otimes \mathbf{M} + \mathbf{M} \otimes \mathbf{A}$$

$$\Delta_0 = \mathbf{B} \otimes \mathbf{M} + \mathbf{M} \otimes \mathbf{B}.$$

Note that by Theorem 2.1, the eigenvector corresponding to λ_c is $z_c = \xi_1 v \otimes \overline{v} + \xi_2 \overline{v} \otimes v$. Therefore, finding λ_c, the quantity that allows us to estimate the critical parameter value α_c, is equivalent to finding the eigenvalue of (2.15) with smallest modulus. One standard approach for computing this eigenvalue is to use an iterative method such as inverse iteration for (2.15). This approach is obviously impractical

for large-scale problems since inverse iteration requires solution of linear systems with coefficient matrix Δ_1, which in this case has order n^2.

To get around this difficulty, we can use properties of Kronecker products to rewrite (2.15) into a linear equation of $n \times n$ matrices. In particular, let $Z \in \mathbb{R}^{n\times n}$ be such that $z = vec(Z)$ (see [23], p. 244). Then it is known (see [23], p. 255) that (2.15) is equivalent to a system of Lyapunov structure

$$\mathbf{M}Z\mathbf{A}^T + \mathbf{A}Z\mathbf{M}^T + \lambda(\mathbf{M}Z\mathbf{B}^T + \mathbf{B}Z\mathbf{M}^T) = 0. \tag{2.16}$$

Therefore, finding λ with smallest modulus for (2.15) is equivalent to finding λ with smallest modulus for (2.16). Because of the relationship between (2.15) and (2.16), we use the same terminology as for (2.15) and (2.16) and refer to λ as an eigenvalue and Z as an eigenvector of (2.16). The following theorem from [29] describes the properties of Z:

Theorem 2.2. *Assume that λ is a real eigenvalue of* (2.15)*. If* (2.13) *has eigenpairs $(\beta i, v)$ and $(-\beta i, \overline{v})$ $(\beta{>}0)$ and no other eigenvalues on the imaginary axis, then* (2.16) *has a real symmetric eigenvector of rank two, namely, $Z = vv^* + \overline{v}v^T$, which is unique up to a scalar factor and is semi-definite, and a unique skew-symmetric eigenvector of rank two, namely, $Z = vv^* - \overline{v}v^T$.*

2.2.2 Inverse Iteration for Problems with Lyapunov Structure

It is suggested in [29] that we should restrict our computation to the real symmetric eigenspace of (2.16). Under this restriction, the eigenvalue of interest, λ_c, is simple.

The corresponding eigenvector, which is symmetric and of rank two, has a natural representation in the form of a truncated eigenvalue decomposition $Z_c = \mathcal{V}\mathcal{D}\mathcal{V}^T$, where $\mathcal{V} \in \mathbb{R}^{n\times 2}$ is orthonormal and $\mathcal{D} \in \mathbb{R}^{2\times 2}$ is diagonal. By Theorem 2.2, $span\{\mathcal{V}\} = span\{v, \overline{v}\}$. Therefore, once we find λ_c and its eigenvector Z_c for (2.16), the rightmost eigenvalues of (2.13) can be found easily by solving the 2×2 problem

$$\mathcal{V}^T(\mathbf{A} + \lambda_c \mathbf{B})\mathcal{V}y = \mu \mathcal{V}^T \mathbf{M}\mathcal{V}y \tag{2.17}$$

The associated eigenvectors are $v = \mathcal{V}y, \overline{v} = \mathcal{V}\overline{y}$. To find the eigenvalue closest to zero for (2.16), the version of inverse iteration outlined in Algorithm 3 can be applied.

Algorithm 3: Inverse iteration for (2.16)

---

1. Given $\mathbf{V}_1 \in \mathbb{R}^n$ with $\|\mathbf{V}_1\|_2 = 1$ and $D_1 = 1$, let $Z^{(1)} = \mathbf{V}_1 D_1 \mathbf{V}_1^T$.
2. For $\ell = 1, 2, \ldots$
   2.1. Compute the eigenvalue approximation[1]

$$\lambda^{(\ell)} = -\frac{trace(\widetilde{A}_\ell^T D_\ell \widetilde{M}_\ell D_\ell + \widetilde{M}_\ell^T D_\ell \widetilde{A}_\ell D_\ell)}{trace(\widetilde{B}_\ell^T D_\ell \widetilde{M}_\ell D_\ell + \widetilde{M}_\ell^T D_\ell \widetilde{B}_\ell D_\ell)} \tag{2.18}$$

   where

$$\widetilde{A}_\ell = \mathbf{V}_\ell^T \mathbf{A}\mathbf{V}_\ell,\ \widetilde{B}_\ell = \mathbf{V}_\ell^T \mathbf{B}\mathbf{V}_\ell,\ \widetilde{M}_\ell = \mathbf{V}_\ell^T \mathbf{M}\mathbf{V}_\ell \tag{2.19}$$

   2.2. If $\left(\lambda^{(\ell)}, Z^{(\ell)}\right)$ is accurate enough, then stop.
   2.3. Else, solve

$$\mathbf{A}Y_\ell \mathbf{M}^T + \mathbf{M}Y_\ell \mathbf{A}^T = \mathbf{B}Z^{(\ell)}\mathbf{M}^T + \mathbf{M}Z^{(\ell)}\mathbf{B}^T \tag{2.20}$$

   in factored form $Y_\ell = \mathbf{V}_{\ell+1} D_{\ell+1} \mathbf{V}_{\ell+1}^T$.
   2.4. Normalize: $D_{\ell+1} \leftarrow D_{\ell+1}/\|D_{\ell+1}\|_F$. Let $Z^{(\ell+1)} = \mathbf{V}_{\ell+1} D_{\ell+1} \mathbf{V}_{\ell+1}^T$.

---

[1]The Rayleigh quotient (2.18) can be derived using a property of Kronecker products (see [23], p. 252, Exercise 25).

If $\mathbf{A}$ is nonsingular, then (2.20) is equivalent to the Lyapunov equation

$$SY_\ell + Y_\ell S^T = \mathbf{A}^{-1} F_\ell \mathbf{A}^{-T} \tag{2.21}$$

where $S = \mathbf{A}^{-1}\mathbf{M}$ and $F_\ell = \mathbf{B}Z^{(\ell)}\mathbf{M}^T + \mathbf{M}Z^{(\ell)}\mathbf{B}^T$. Let $rank\left(Z^{(\ell)}\right) = d_\ell$; it is reasonable to assume that $d_\ell \ll n$ (see [2, 31]). The right-hand side of (2.21) can be represented by its truncated eigenvalue decomposition

$$\mathbf{A}^{-1} F_\ell \mathbf{A}^{-T} = P_\ell C_\ell P_\ell^T \tag{2.22}$$

which has rank at most $2d_\ell$ and is easy to compute.[2] Since we assume that $\mathbf{M}$ is nonsingular and the point $(\overline{u}_0, \alpha_0)$ is in the stable regime, all the eigenvalues of S lie in the left half of the complex plane. This guarantees that (2.21) has a unique solution (see [1], Chapter 6).

Theorem 2.2 implies that Z_c has rank 2, so when $Z^{(\ell)}$ has converged, the right-hand side of (2.21), namely (2.22), has rank 4. For efficient computation of (2.21), we would like to work with $d_\ell = 2$. However, in the first few iterations, when $Z^{(\ell)}$ has not converged yet, d_ℓ can be much larger than 2 (although $d_\ell \ll n$). A rank-reduction procedure is introduced in [29] to guarantee that $rank\left(Z^{(\ell)}\right)$ is fixed and small. Before step 2.2 in Algorithm 3, we project the eigenvalue problem (2.16) onto the subspace spanned by the columns of $\mathbf{V}_\ell$. This leads to the $d_\ell \times d_\ell$ eigenvalue

[2] Let $T = \mathbf{A}^{-1}\mathbf{B}$; then

$$\mathbf{A}^{-1} F_\ell \mathbf{A}^{-T} = \left(\frac{\sqrt{2}}{2}\left[T\mathbf{V}_\ell + S\mathbf{V}_\ell, T\mathbf{V}_\ell - S\mathbf{V}_\ell\right]\right) \begin{bmatrix} D_\ell & \\ & -D_\ell \end{bmatrix} \left(\frac{\sqrt{2}}{2}\left[T\mathbf{V}_\ell + S\mathbf{V}_\ell, T\mathbf{V}_\ell - S\mathbf{V}_\ell\right]\right)^T.$$

problem

$$\widetilde{M}_\ell \widetilde{Z} \widetilde{A}_\ell^T + \widetilde{A}_\ell \widetilde{Z} \widetilde{M}_\ell^T + \widetilde{\lambda} \left(\widetilde{M}_\ell \widetilde{Z}_\ell \widetilde{B}_\ell^T + \widetilde{B}_\ell \widetilde{Z}_\ell \widetilde{M}_\ell^T \right) = 0 \tag{2.23}$$

where $\widetilde{A}_\ell$, $\widetilde{B}_\ell$, $\widetilde{M}_\ell$ are computed in (2.19). For $d_\ell \ll n$, the eigenvalue $\widetilde{\lambda}_c$ of (2.23) with smallest modulus and it corresponding eigenvector $\widetilde{Z}_c$ can be computed using Algorithm 3 with a direct Lyapunov solver (see [4, 22]) in step 2.3. According to Theorem 2.2, $\widetilde{Z}_c$ has rank 2. Let the eigenvalue decomposition of $\widetilde{Z}_c$ be $\widetilde{\mathcal{V}} \widetilde{\mathcal{D}} \widetilde{\mathcal{V}}^T$, where $\widetilde{\mathcal{V}} \in \mathbb{R}^{d_\ell \times 2}$ and $\widetilde{\mathcal{D}} \in \mathbb{R}^{2 \times 2}$. We update the eigenvector $Z^{(\ell)} = \mathbf{V}_\ell D_\ell \mathbf{V}_\ell^T$ by $\left(\mathbf{V}_\ell \widetilde{\mathcal{V}} \right) \widetilde{\mathcal{D}} \left(\mathbf{V}_\ell \widetilde{\mathcal{V}} \right)^T$. The new eigenvector has rank 2 and it forces the residual of (2.16) to be orthogonal to $\mathbf{V}_\ell$. With the rank-reduction procedure, the right-hand side of (2.21) will be of rank 2 in the first iteration and of rank 4 in all subsequent iterations, which is desirable for the Lyapunov solvers. The modified Lyapunov inverse iteration for (2.16) is given in Algorithm 4 and we refer to it as Lyapunov inverse iteration.

Algorithm 4: Lyapunov inverse iteration for (2.16)

1. Given $\mathbf{V}_1 \in \mathbb{R}^n$ with $\|\mathbf{V}_1\|_2 = 1$.
2. For $\ell = 1, 2, \cdots$
   - 2.1. Compute (2.19), and solve for the eigenvalue $\widetilde{\lambda}_c$ of (2.23) closest to zero and its eigenvector $\widetilde{Z}_c = \widetilde{\mathcal{V}} \widetilde{\mathcal{D}} \widetilde{\mathcal{V}}^T$.
   - 2.2. Set $Z^{(\ell)} = \mathcal{V}_\ell \widetilde{\mathcal{D}} \mathcal{V}_\ell^T$ and $\lambda^{(\ell)} = \widetilde{\lambda}_c$, where $\mathcal{V}_\ell = \mathbf{V}_\ell \widetilde{\mathcal{V}}$.
   - 2.3. If $\left(\lambda^{(\ell)}, Z^{(\ell)} \right)$ is accurate enough, then stop.
   - 2.4. Else, solve for Y_ℓ from

$$S Y_\ell + Y_\ell S^T = P_\ell C_\ell P_\ell^T \tag{2.24}$$

   in factored form: $Y_\ell = \mathbf{V}_{\ell+1} D_{\ell+1} \mathbf{V}_{\ell+1}^T$.

Chapter 3

Lyapunov Inverse Iteration for Identifying Hopf Bifurcations in Models of Incompressible Flow

As shown in [29], the critical parameter value α_c at which stability of the steady-state solution is lost can be estimated by solving an eigenvalue problem in the form of a Lyapunov equation, namely, (2.16), for its eigenvalue with smallest modulus. The advantage of this method is that it avoids the computation of the rightmost eigenvalue of (2.2), for which no robust method exists. To compute this eigenvalue, a version of inverse iteration called Lyapunov inverse iteration (see Algorithm 4 of section 2.2) is also proposed in [29]. The main cost of this method is solving a large Lyapunov equation (2.24) at each iteration. The rightmost eigenvalue of (2.2) at the critical point and its corresponding eigenvector can be obtained as by-products.

The aims of this chapter are: (i) to further understand and refine the method discussed in [29] to make it more efficient and reliable, (ii) to test it on more challenging examples arising in fluid dynamics, and (iii) to provide a discussion of the efficiency of large-scale Lyapunov solvers arising from this approach.

Consider a special case of (2.1), the Navier-Stokes equations governing viscous

incompressible flow,

$$\begin{aligned} u_t &= \nu\nabla^2 u - u\cdot\nabla u - \nabla p \\ 0 &= \nabla\cdot u, \end{aligned} \tag{3.1}$$

subject to appropriate boundary conditions, where ν is the kinematic viscosity, u is the velocity and p is the pressure. The viscosity ν is a natural candidate for α. In the literature, properties of a flow are usually characterized by the Reynolds number (denoted by Re), a dimensionless quantity proportional to $\frac{1}{\nu}$. For convenience in our exposition, we will sometimes refer to the Reynolds number as the parameter of interest instead of the viscosity. Div-stable mixed finite element discretization of (3.1) gives rise to the following Jacobian matrix and mass matrix [13]

$$\mathbf{A} = \begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix},\ \mathbf{M} = \begin{bmatrix} -G & 0 \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{n\times n} \tag{3.2}$$

where $n = n_u + n_p$, $n_u > n_p$, $F \in \mathbb{R}^{n_u\times n_u}$, $B \in \mathbb{R}^{n_p\times n_u}$, $G \in \mathbb{R}^{n_u\times n_u}$ is symmetric positive definite. Matrices F, B, G are sparse and n is usually large. In this chapter, we apply the method proposed in [29] to detect the Hopf point at which a steady-state solution of (3.1) loses its stability.

The plan for the rest of the chapter is as follows. In section 3.1, we discuss a block Krylov method for solving large-scale Lyapunov equations with low-rank right-hand side, and we propose an efficient way to truncate the computed solution. In section 3.2, we propose an inverse iteration with inexact Lyapunov solvers, which is based on the ideas in [34]. In section 3.3, the method proposed in section 3.2 is applied to detect Hopf bifurcation in two incompressible flows and numerical

results are presented; in addition, alternative Lyapunov solvers are discussed and compared with the Krylov method of section 3.1. Finally, in section 3.4, we make some concluding observations.

3.1 A Block Krylov Lyapunov Solver

In this section, we discuss the block Krylov method for solving the Lyapunov equation

$$SY + YS^T = PCP^T \tag{3.3}$$

where $S = \mathbf{A}^{-1}\mathbf{M} \in \mathbb{R}^{n\times n}$, $P \in \mathbb{R}^{n\times p}$ orthonormal, and $C \in \mathbb{R}^{p\times p}$ diagonal, with $p \ll n$. (In Algorithm 4, the right-hand side of the Lyapunov equation (2.24) has rank $p = 2$ in the first iteration and rank $p = 4$ in all subsequent iterations.) Let K be a d-dimensional subspace of $\mathbb{R}^n$ and let $\mathbf{V} \in \mathbb{R}^{n\times d}$ be an orthonormal matrix whose columns form a basis of K. Projection methods for (3.3) seek an approximate solution of the form $Y^{approx}(Q) = \mathbf{V}Q\mathbf{V}^T$ with $Q \in \mathbb{R}^{d\times d}$ by imposing the so-called Galerkin condition, *i.e.*, the residual $R(Q) = SY^{approx}(Q) + Y^{approx}(Q)S^T - PCP^T$ of (3.3) must satisfy

$$\langle Z, R(Q)\rangle = tr\left(ZR(Q)^T\right) = 0 \tag{3.4}$$

for any matrix Z of the form $\mathbf{V}\mathcal{G}\mathbf{V}^T$ with $\mathcal{G} \in \mathbb{R}^{d\times d}$ (see [39]). The only Q that satisfies this condition is the solution to the projected problem (see [39])

$$\left(\mathbf{V}^T S\mathbf{V}\right)Q + Q\left(\mathbf{V}^T S\mathbf{V}\right)^T = \left(\mathbf{V}^T P\right)C\left(\mathbf{V}^T P\right)^T \tag{3.5}$$

In the block Krylov method (see [24, 39]), the subspace K is chosen to be

$$\mathcal{K}_m(S, P) = span\left\{P, SP, S^2P, \cdots, S^{m-1}P\right\} \tag{3.6}$$

(The dimension of $\mathcal{K}_m(S, P)$ is $d = mp$.) One theoretical motivation for selecting such a subspace is that if all the eigenvalues of S lie in the left half of the complex plane, then the analytic solution of (3.3) can be expressed as

$$-\int_0^\infty \exp(St)PCP^T \exp(S^Tt)\, dt$$

(see [1], Chapter 6). We use the block Arnoldi method to compute an orthonormal basis for $\mathcal{K}_m(S, P)$. Similar to the standard Arnoldi method, the block Arnoldi process computes a decomposition

$$S\mathbf{V} = \mathbf{V}H_m + V_{m+1}H_{m+1,m}E_m^T \tag{3.7}$$

where the columns of $\mathbf{V} = [V_1, \cdots, V_m] \in \mathbb{R}^{n\times mp}$ and V_{m+1} form an orthonormal basis for $\mathcal{K}_{m+1}(S, P)$, $H_m \in \mathbb{R}^{mp\times mp}$ is a block upper-Hessenberg matrix with $p \times p$ blocks $H_{i,j}$, and $E_m \in \mathbb{R}^{mp\times p}$ is the last p columns of the identity matrix of order mp. By the Arnoldi relationship (3.7), the projected problem (3.5) is

$$H_mQ + QH_m^T = \begin{bmatrix} C & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} = \widetilde{C} \tag{3.8}$$

which, assuming $mp \ll n$, can be solved by direct methods. An algorithmic form of the block Krylov method for solving (3.3) is given by Algorithm 5.

Algorithm 5: The block Krylov method for (3.3)

1. Given a tolerance τ. Let $V_1 = \mathbf{V} = P$.
2. For $m = 1, 2, \cdots$
   - 2.1. $W = SV_m$.
     for $i = 1, \ldots, m$
     $H_{i,m} \leftarrow V_i^T W$;
     $W \leftarrow W - V_i H_{i,m}$.
   - 2.2. Solve the smaller Lyapunov equation (3.8) where $H_m = [H_{j,k}]_{j,k=1}^m$.
   - 2.3. Compute the reduced QR factorization of W: $W = V_{m+1} H_{m+1,m}$.
   - 2.4. Compute the residual norm $\|R(Q)\|_F$.
   - 2.5. If $\|R(Q)\|_F < \tau$, then stop.
   - 2.6. Else, $\mathbf{V} \leftarrow [\mathbf{V}, V_{m+1}]$.

We outline some of the computational issues associated with this algorithm. Since $S = \mathbf{A}^{-1}\mathbf{M}$, in step 2.1, we need to solve p linear systems of the form

$$\mathbf{A}x = \mathbf{M}y \tag{3.9}$$

for x. Notice that we do not need to form the approximate solution $Y^{approx}(Q) = \mathbf{V}Q\mathbf{V}^T$ explicitly. Instead, only the factors $\mathbf{V}$ and Q are stored. To compute the residual norm $\|R(Q)\|_F$, first notice that for any symmetric Q,

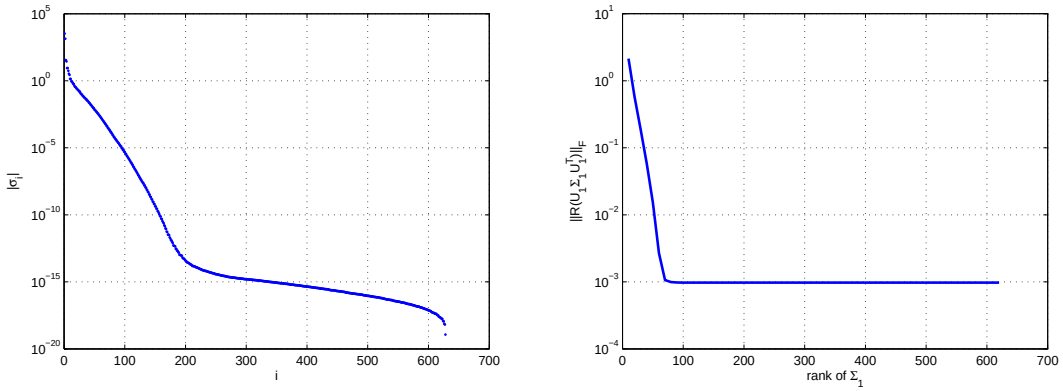
$$R(Q) = [\mathbf{V}, V_{m+1}] \begin{bmatrix} H_m Q + Q H_m^T - \widetilde{C} & Q E_m H_{m+1,m}^T \\ H_{m+1,m} E_m^T Q & 0 \end{bmatrix} [\mathbf{V}, V_{m+1}]^T \tag{3.10}$$

(see [24]). By (3.8) and (3.10), $\|R(Q)\|_F = \sqrt{2}\left\|Q E_m H_{m+1,m}^T\right\|_F$ which is cheap to compute. Let $Q = U\Sigma U^T$ be the eigenvalue decomposition of Q where $\Sigma = diag(\sigma_1, \sigma_2, \cdots, \sigma_m)$ holds the eigenvalues of Q, where the moduli are in decreasing order. The computed solution $Y^{approx}(Q)$ can usually be truncated to a (much)

lower rank without affecting the residual norm:

$$Y^{approx}(Q) = (\mathbf{V}U)\Sigma(\mathbf{V}U)^T = (V\,[U_1, U_2]) \begin{bmatrix} \Sigma_1 & \\ & \Sigma_2 \end{bmatrix} (\mathbf{V}\,[U_1, U_2])^T \approx Y^{approx}\left(U_1\Sigma_1 U_1^T\right). \tag{3.11}$$

In order to do this, we increase the rank of Σ_1 until the residual norm of the truncated solution, $\|R(U_1\Sigma_1 U_1^T)\|_F$, is smaller than a prescribed tolerance τ. For example, consider the Lyapunov equation arising from the first iteration of Algorithm 4, when applied to the flow over an obstacle (details of this example are given in section 5.2). Let the tolerance $\tau = 10^{-3}$. The solution computed by Algorithm 5 has rank 628 and can be truncated to rank 80 without significantly affecting its accuracy. Figure 3.1(a) shows the decay of eigenvalues of Q, and Figure 3.1(b) depicts the residual norm of truncated solutions corresponding to various choices of Σ_1.



(a) Decay of the eigenvalues of Q

(b) Residual norm for different ranks of truncation

Figure 3.1: Low-rank approximation of the solution to the Lyapunov equation ($n = 37168$)

In our experiments, we have observed that when applying Algorithm 4 to problems arising from fluid mechanics, solving the Lyapunov equation (3.3) accurately can be quite expensive, especially at the early stages of the computation when the eigenvector $Z^{(\ell)}$ has not converged yet. In the next section, we will show that it is

in fact not necessary to solve (3.3) accurately in the first few iterations of Algorithm 4.

Variants of the standard Krylov method described here, for example, the Extended Krylov Subspace Method [42] and the Rational Krylov Subspace Method [12], lead to different choices of the subspace K. Some results for the alternative methods will be given in section 3.3.

Remark. Necessary and sufficient condition for the projected Lyapunov equation (3.8) to have a unique solution is that $\theta_i + \theta_j \neq 0$ where (θ_i, θ_j) is any pair of eigenvalues of H_m. To guarantee this, in the literature on projection methods for Lyapunov equations, it is common to require that the field of values of S lies in one half of the complex plane (the imaginary axis not included). This is a rather strong condition which is not satisfied by matrices in our numerical experiments described in section 3.3. (Instead, they only satisfy a weaker condition that all their eigenvalues lie in the left half of the complex plane.) Thus, it is conceivable that the solution to (3.8) does not exist or is not unique, which will lead to a breakdown of the projection method. We never encountered this difficulty in our experiments.

3.2 Inexact Lyapunov Inverse Iteration

In this section, we first review the main results from the previous work of Robbé *et al.* [34] on inexact inverse iteration and based on their idea, we propose an inexact inverse iteration for solving the eigenvalue problem (2.16). Suppose that a cluster ($k \ll n$) of eigenvalues of $\mathcal{A} \in \mathbb{R}^{n \times n}$ near a shift s is wanted. The standard approach

for this problem is inverse iteration, which requires the solution of k linear systems

$$(\mathcal{A} - sI)X_i = X_{i-1} \tag{3.12}$$

at each step. Solving (3.12) exactly can be very challenging if n is large, which is typical when $\mathcal{A}$ arises from discretization of two- or three-dimensional PDEs. Therefore, the system (3.12) is often solved inexactly using iterative methods. This approach is referred to as an *inner-outer* iterative method: the inner iteration refers to the iterative solution of (3.12) and the outer iteration is inverse iteration for eigenvalues. For simplicity, let $k = 1$ and $s = 0$, that is, suppose we are looking for the eigenvalue closest to zero. The inexact inverse iteration in this case is as outlined in Algorithm 6.

Algorithm 6: Inexact inverse iteration

1. Given a tolerance $\delta > 0$ and the starting guess $z^{(1)}$ with $\left\|z^{(1)}\right\|_2 = 1$.
2. For $\ell = 1, 2, \cdots$
   - 2.1. Compute the eigenvalue estimate: $\lambda^{(\ell)} = \left(z^{(\ell)}\right)^T \mathcal{A} z^{(\ell)}$.
   - 2.2. Set $r_\ell^{eig} = \mathcal{A}z^{(\ell)} - \lambda^{(\ell)} z^{(\ell)}$ and test convergence.
   - 2.3. Compute an approximate solution y_ℓ^{approx} to $\mathcal{A}y_\ell = z^{(\ell)}$ such that $r_\ell^{lin} = \mathcal{A}y_\ell^{approx} - z^{(\ell)}$ with
   $$\|r_\ell^{lin}\|_2 < \delta \|r_\ell^{eig}\|_2. \tag{3.13}$$
   - 2.4. Normalize: $z^{(\ell+1)} = y_\ell^{approx} / \|y_\ell^{approx}\|_2$

Since δ is fixed, the stopping criterion (3.13) implies the following: at the early stage of the eigenvalue computation, when $\|r_\ell^{eig}\|_2$ is still large, the inner iteration does not need to be very accurate either; as $\left(\lambda^{(\ell)}, z^{(\ell)}\right)$ converges to the true solution (*i.e.*, $\|r_\ell^{eig}\|_2$ gets smaller), (3.12) will be solved more and more accurately. It was

shown in [34] that with this strategy, the number of inner iterations will not increase as the outer iteration proceeds.

We have a similar situation here: we want to compute the eigenvalue of (2.16) closest to zero using Algorithm 4, which requires the solution of equation (2.24) at each step. Note that $\|z\|_2 = \|Z\|_F$ if $z = vec(Z)$. Moreover, for $\mathbf{A}$ nonsingular, (2.16) is equivalent to

$$SZ + ZS^T + \lambda(SZT^T + TZS^T) = 0 \tag{3.14}$$

where $S = \mathbf{A}^{-1}\mathbf{M}$ and $T = \mathbf{A}^{-1}\mathbf{B}$. Therefore, in Algorithm 4, the stopping criterion

$$\|R_\ell^{lyap}\|_F < \delta \|R_\ell^{eig}\|_F \tag{3.15}$$

is used for the inner iteration (*e.g.*, Algorithm 5) for (2.24), where $R_\ell^{eig} = SZ^{(\ell)} + Z^{(\ell)}S^T + \lambda^{(\ell)}\left(SZ^{(\ell)}T^T + TZ^{(\ell)}S^T\right)$ and $R_\ell^{lyap} = SY_\ell^{approx} + Y_\ell^{approx}S^T - P_\ell C_\ell P_\ell^T$. Based on Algorithms 4 and 6, we propose a version of inexact Lyapunov inverse iteration for solving (2.16), which is outlined in Algorithm 7.

Algorithm 7: Inexact Lyapunov inverse iteration for (2.16)

1. Given $\mathbf{V}_1 \in \mathbb{R}^n$ with $\|\mathbf{V}_1\|_2 = 1$ and $\delta > 0$.
2. For $\ell = 1, 2, \ldots$
   - 2.1. Compute (2.19), and solve for the eigenvalue $\widetilde{\lambda}_c$ of (2.23) closest to zero and its eigenvector $\widetilde{Z}_c = \widetilde{\mathcal{V}}\widetilde{\mathcal{D}}\widetilde{\mathcal{V}}^T$.
   - 2.2. Set $Z^{(\ell)} = \mathcal{V}_\ell \widetilde{D} \mathcal{V}_\ell^T$ and $\lambda^{(\ell)} = \widetilde{\lambda}_c$, where $\mathcal{V}_\ell = \mathbf{V}_\ell \widetilde{\mathcal{V}}$.
   - 2.3. Compute $\|R_\ell^{eig}\|_F$ and test convergence.
   - 2.4. Compute an approximate solution $Y_\ell^{approx} = \mathbf{V}_{\ell+1} D_{j+1} \mathbf{V}_{\ell+1}^T$ for (2.24) such that $\|R_\ell^{lyap}\|_F < \delta \|R_\ell^{eig}\|_F$.

Remark. An alternative choice of the pair of residuals would be $\mathcal{R}_\ell^{eig} =$

$\mathbf{M}Z^{(\ell)}\mathbf{A}^T + \mathbf{A}Z^{(\ell)}\mathbf{M}^T + \lambda^{(\ell)}(\mathbf{M}Z^{(\ell)}\mathbf{B}^T + \mathbf{B}Z^{(\ell)}\mathbf{M}^T)$ and $\mathcal{R}_\ell^{lyap} = \mathbf{A}Y_\ell\mathbf{M}^T + \mathbf{M}Y_\ell\mathbf{A}^T - \left(\mathbf{B}Z^{(\ell)}\mathbf{M}^T + \mathbf{M}Z^{(\ell)}\mathbf{B}^T\right)$, which are the residuals of (2.16) and (2.20), respectively. We prefer the choice used in Algorithm 7 because of cost considerations. $\|R_\ell^{eig}\|_F$ is available at almost no cost due to (3.10), and since $Z^{(\ell)} = \mathcal{V}_\ell\widetilde{D}\mathcal{V}_\ell^T$ has rank two, R_ℓ^{eig} has rank four and the dominant cost of computing $\|R_\ell^{eig}\|_F$ is the solution of two systems with coefficient matrix $\mathbf{A}$ to compute $T\mathcal{V}_\ell$.[1] (In order to get $\|R_\ell^{eig}\|_F$, we need to compute $S\mathcal{V}_\ell$ as well. By using the Arnoldi decomposition (3.7), this can be computed cheaply without any extra solves with $\mathbf{A}$.) In contrast, although it is trivial to compute the Frobenius norm of the rank-four $\mathcal{R}_\ell^{eig}$, it can be very expensive to evaluate $\|\mathcal{R}_\ell^{lyap}\|_F$: by (3.10) and the relation $\mathcal{R}_\ell^{lyap} = \mathbf{A}R_\ell^{lyap}\mathbf{A}^T$, computing $\|\mathcal{R}_\ell^{lyap}\|_F$ at the m^{th} step of Algorithm 5 applied to (2.24) requires $p(m+1)$ matrix-vector products with $\mathbf{A}$, where p is the rank of the right-hand side of (2.24). If a large number of Arnoldi steps is needed, which is indeed the case in our numerical experiments, then monitoring $\|\mathcal{R}_\ell^{lyap}\|_F$ instead of $\|R_\ell^{lyap}\|_F$ will be much more expensive.

3.3 Numerical Results

In this section, we apply Algorithm 7 to two 2-dimensional models of incompressible flows that lose stability because of Hopf bifurcation, namely, driven-cavity flow and flow over an obstacle. The numerical results support the theory of [29] and show that the algorithm we propose is robust.

In the previous sections, we always assumed that the mass matrix $\mathbf{M}$ is nonsin-

[1] In order to solve (2.24), $\mathbf{A}$ has been pre-factored or a preconditioner for it has been computed.

gular. However, as given by (3.2), the mass matrix in our examples is singular. This implies that (2.2) has an *infinite eigenvalue* (*i.e.*, the eigenvalue that corresponds to the zero eigenvalue of S) of multiplicity $2n_p$ (see [8]). As shown in [8], however, replacement of $\mathbf{M}$ with the nonsingular, shifted mass matrix

$$\mathbf{M}_\eta = \begin{bmatrix} -G & \eta B^T \\ \eta B & 0 \end{bmatrix} \tag{3.16}$$

maps the infinite eigenvalue of (2.2) to η^{-1} and leaves the finite ones unchanged. With a proper choice of η, the rightmost eigenvalue(s) of (2.2) will not be changed, which means that stability analysis will not be affected. In our computations, we use the shifted mass matrix (3.16) with $\eta = -10^{-2}$ instead of the $\mathbf{M}$ given in (3.2). The infinite eigenvalues of (2.2) are mapped to -10^2, which is well away from its rightmost eigenvalues. In addition, the matrix $\mathbf{B} = \frac{d\mathcal{J}}{d\nu}(\nu_0)$ was approximated using a forward difference approximation.

All numerical results were obtained using Matlab 7.8.0 (R2009a), on a PC with an Intel Core i7 720QM processor, and 4 GB of RAM.

3.3.1 Example 1: Driven-Cavity Flow

This is a classic test problem used in fluid dynamics, a model of the flow in a unit-square cavity with the lid moving from left to right. We use the software package IFISS (see [13]) to compute the steady-state solution of (3.1). The left plot of Figure 3.2 shows exponentially distributed streamlines of a steady solution. Studies of the critical Reynolds number Re_c for this problem show it to be around 8000. (For example, the reported value is 7998.5 in [16], 7960 in [18], between 8017.6

and 8018.8 in [3], and between 8000 and 8050 within less than 1% error in [5].) The rightmost eigenvalues at the critical Reynolds number are also provided in [16] ($\pm\beta i \approx \pm 2.8356i$) and [18] ($\pm\beta i \approx \pm 2.837i$). The right plot of Figure 3.2 shows the eigenvalues of $\mathcal{J}x = \mu \mathbf{M}x$ at $Re = 8076$, a value slightly larger than the critical value Re_c. As is clearly seen, there are many complex eigenvalues near the imaginary axis, and, in fact, it is a very difficult problem to find out precisely which eigenpair crosses the imaginary axis to cause the loss of stability.

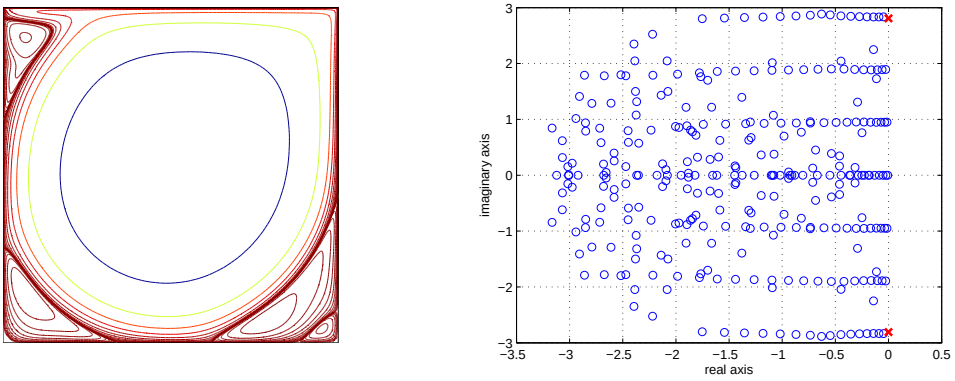


Figure 3.2: Driven-cavity flow (256×256 mesh). Left: Exponentially distributed streamlines at $Re = 7500$. Right: The 300 eigenvalue of (2.2) with smallest modulus at $Re = 8076$ (the crosses denote the rightmost eigenvalues)

We use a Q_2-Q_1 mixed finite element discretization and three meshes: 64×64 ($n = 9539$), 128×128 ($n = 37507$), and 256×256 ($n = 148739$). Algorithm 7 is tested on the three problems arising from the three meshes of discretization, with tests for three choices $\delta = 1,\ 10^{-1}$ and 10^{-2} in (3.15). Let the Reynolds number at the starting point, Re_0, be about 250 smaller than its critical value, Re_c. The goal of our tests is to find out whether Algorithm 7 is able to approximate the difference λ_c between the two viscosities $\nu_0 = \frac{1}{Re_0}$ and $\nu_c = \frac{1}{Re_c}$ and in turn, give us

a good estimate of Re_c.[2] The computational results for the finest mesh are reported in Table 3.2. $Re^{(\ell)}$ denotes the estimated value of Re_c, $\mu^{(\ell)}$ denotes the estimated βi, $r_\ell = \left(\mathbf{A} + \lambda^{(\ell)}\mathbf{B}\right) x^{(\ell)} - \mu^{(\ell)}\mathbf{M}x^{(\ell)}$ is the residual of (2.13), and R_ℓ^{lyap}, R_ℓ^{eig} are defined in the previous section. In addition, d_ℓ is the rank of the solution of (2.24) before truncation, and k_ℓ is the rank after truncation. (A summary of the symbols used in Table 3.2 and their definitions is given in Table 3.1.) The main cost of each iteration is approximately d_ℓ solves of linear systems with coefficient matrix $\mathbf{A}$. The computation terminates when $\|r_\ell\|_2 < 10^{-9}$ is satisfied. In the first iteration, when a real, symmetric and rank-one matrix $\mathbf{V}_1\mathbf{V}_1^T$ ($\mathbf{V}_1$ is a random vector in $\mathbb{R}^n$) is used as the eigenvector estimate of (2.16), the eigenvalue estimate $\lambda^{(1)}$ is quite far away from its true value λ_c, causing the estimated critical Reynolds number to be nonphysical (-219). However, starting from the second iteration, $\lambda^{(\ell)}$ converges rapidly to its true value. A fairly large Krylov subspace is needed to solve the Lyapunov equations, even when the tolerance is quite mild ($\|R_\ell^{lyap}\|_F < \|R_\ell^{eig}\|_F$). Computational results for the two coarser meshes can be found in the Appendix and the same trend can be observed there.

As observed in Chapter 2, a commonly used method to locate the first Hopf point is to compute the rightmost eigenvalues of (2.2) for a set of points with increasing Reynolds numbers on the solution path $\mathcal{S}$, until a critical value is reached at which the real part of the rightmost eigenvalues becomes positive. We follow this approach to verify the results given by Algorithm 7. The details are as follows:

[2]Let $\lambda_c = \nu_c - \nu_0$; then once λ_c is approximated by Algorithm 7, Re_c can be estimated by $\frac{1}{\nu_0 + \lambda_c}$.

Table 3.1: Notation for Algorithm 7

Symbol	Definition
$Re^{(\ell)}$	the estimate of Re_c, *i.e.*, the critical Reynolds number
$\mu^{(\ell)}$, $x^{(\ell)}$	the estimated rightmost eigenvalue of (2.2) at the critical point and its corresponding eigenvector, respectively
$\lambda^{(\ell)}$, $Z^{(\ell)}$	the estimated eigenvalue of (3.14) with smallest modulus, λ_c, and its associated eigenvector Z_c, respectively
r_ℓ	$\left(\mathbf{A}+\lambda^{(\ell)}\mathbf{B}\right)x^{(\ell)}-\mu^{(\ell)}\mathbf{M}x^{(\ell)}$, *i.e.*, the residual of (2.13)
R_ℓ^{eig}	$SZ^{(\ell)}+Z^{(\ell)}S^T+\lambda^{(\ell)}\left(SZ^{(\ell)}T^T+TZ^{(\ell)}S^T\right)$, *i.e.*, the residual of (3.14)
Y_ℓ^{approx}	the approximate solution to (2.24)
R_ℓ^{lyap}	$SY_\ell^{approx}+Y_\ell^{approx}S^T-P_\ell C_\ell P_\ell^T$, *i.e.*, the residual of (2.24)
d_ℓ, k_ℓ	ranks of Y_ℓ^{approx} before and after the truncation (3.11), respectively

for each point in the set, we compute the 250 eigenvalues with smallest modulus for (2.2) using Matlab function 'eigs' (with other parameters set to default values), which implements the *implicitly restarted Arnoldi* (IRA) method [43]. For the finest mesh, the critical Reynolds number found by this method is between 8075 and 8076, and the rightmost eigenvalues are $\pm\beta i \approx \pm 2.80905i$. This shows that Algorithm 7 yields good estimates of Re_c and βi. The number 250 was obtained by trial and error. When only 200 eigenvalues with smallest modulus were computed, we could not find the rightmost eigenvalues.

Remark. Our goal is to have a robust method that detects instability without computing many eigenvalues, since we do not know in general how many eigenvalues need be computed to ensure that the rightmost ones have been found. It is also not straightforward to evaluate the cost of the IRA method when it is used to generate a set of eigenvalues in this way, because this cost is highly dependent on how various parameters are chosen. Consequently, we do not make a detailed cost comparison of the two methods. For the particular choice of parameters we made, *i.e.*, computing

Table 3.2: Algorithm 7 applied to driven-cavity flow (256×256 mesh, $Re_0 = 7800$) for $\delta = 1,\ 10^{-1},\ 10^{-2}$ in (3.15)

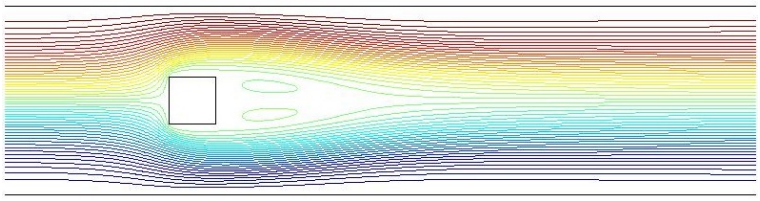
ℓ	$Re^{(\ell)}$	$\mu^{(\ell)}$	$\|r_\ell\|_2$	$\|R_\ell^{eig}\|_F$	$\|R_\ell^{lyap}\|_F$	d_ℓ	k_ℓ
			$\delta = 1$				
1	-219	2.64515e-12	1.29459e-01	4.94209e+1	4.86367e+1	322	90
2	8014	2.81408i	1.72108e-06	1.52388e-2	1.48458e-2	424	160
3	8080	2.80919i	7.33553e-08	4.42196e-4	3.87001e-4	444	160
4	8077	2.80960i	3.43710e-09	1.61958e-5	1.58346e-5	448	170
5	8077	2.80960i	1.04455e-10	5.90803e-7	—	—	—
					Total:	1638	
			$\delta = 10^{-1}$				
1	-219	2.64515e-12	1.29459e-01	4.94209e+1	4.79199e+0	510	120
2	8173	2.81562i	2.54877e-06	2.57187e-2	2.43960e-3	536	190
3	8083	2.80915i	7.32265e-08	3.57523e-4	3.37686e-5	552	210
4	8077	2.80960i	4.06199e-09	2.07058e-5	2.00807e-6	532	210
5	8077	2.80960i	1.53027e-10	8.23020e-7	—	—	—
					Total:	2130	
			$\delta = 10^{-2}$				
1	-219	2.64515e-12	1.29459e-01	4.94209e+1	4.88503e-1	914	190
2	8291	2.81758i	2.85886e-06	3.19166e-2	3.14783e-4	724	260
3	8082	2.80908i	7.05097e-08	4.49294e-4	4.32428e-6	728	270
4	8077	2.80960i	5.23912e-09	1.97292e-5	1.92897e-7	724	260
5	8077	2.80960i	1.51600e-10	6.70318e-7	—	—	—
					Total:	3090	

the 250 eigenvalues with smallest modulus using 'eigs' with default setting, at each Reynolds number in the set, the eigenvalue computation requires the solution of at least 500 linear systems with coefficient matrix $\mathcal{J}$, and typically many more. In our experience, locating Re_c by monitoring the rightmost eigenvalues along $\mathcal{S}$ is much more expensive than Algorithm 7 with $\delta = 1$.

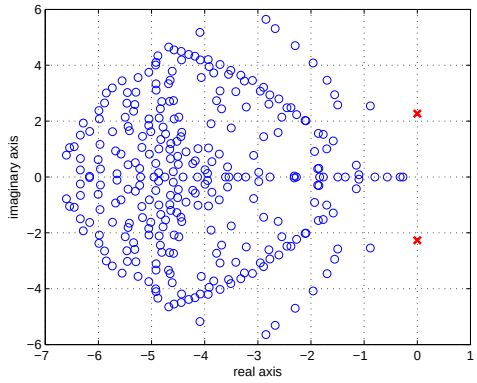
3.3.2 Example 2: Flow over an Obstacle

This example represents flow in a channel (dimension: 2×8) with a square obstacle (dimension: 0.5×0.5) in it. (In this case, the Reynolds number is defined to be

$\frac{2}{\nu}$.) A Poiseuille flow profile is imposed on the inflow boundary, and a no-flow (zero velocity) condition is imposed on the walls. A Neumann condition is applied at the outflow boundary and automatically sets the mean outflow pressure to zero (see [13] for details). Again we use IFISS to compute the steady-state solution. Uniformly distributed streamlines of the steady solution are plotted in Figure 3.3(a). As in the previous example, we use Q_2-Q_1 mixed finite element discretization and apply Algorithm 7 (with $\delta = 1, 10^{-1}, 10^{-2}$) on three meshes: 32×128 ($n = 9512$), 64×256 ($n = 37168$) and 128×512 ($n = 146912$).



(a) Uniformly distributed streamlines at $Re = 350$



(b) The 300 eigenvalues of $\mathcal{J}x = \mu \mathbf{M}x$ with smallest modulus at $Re = 373$ (the crosses denote the rightmost eigenvalues)

Figure 3.3: Flow over an obstacle (128×512 mesh)

We choose Re_0 to be 50 smaller than the critical value Re_c. The computational results for the finest mesh are reported in Table 3.3 (see Table 3.1 for notation). Results given by the IRA method (see Example 1) are the following: $372 \leq Re_c \leq$

373 and $\pm\beta i \approx \pm 2.26578i$. The 300 eigenvalues with smallest modulus at a Reynolds number slightly larger than Re_c are plotted in Figure 3.3(b). As in the previous example, our algorithm gives good estimates of Re_c and βi. This problem has significantly fewer eigenvalues near the imaginary axis, and the Krylov subspaces needed for the Lyapunov solves are also significantly smaller than for the cavity problem. Computational results for the other two meshes can be found in the Appendix.

Remark. Note that $\mathbf{A} + \lambda_c \mathbf{B}$ is only a linear approximation of the true Jacobian matrix $\mathcal{J}(\alpha_c)$. Therefore, the true eigenvalue of (2.16), (2.15) and (3.14) with smallest modulus is $\lambda_c + \epsilon$ where ϵ is a small error. The size of this error, $|\epsilon|$, depends on the distance between the chosen starting point $(\overline{u}_0, \alpha_0)$ and the critical point $(\overline{u}_c, \alpha_c)$: the closer $(\overline{u}_c, \alpha_c)$ is to $(\overline{u}_0, \alpha_0)$, the better $\mathbf{A} + \lambda_c \mathbf{B}$ approximates $\mathcal{J}(\alpha_c)$ and the smaller $|\epsilon|$ gets.

3.3.3 Discussion of Lyapunov Solvers

As observed above, the efficiency of Algorithm 7 depends largely on the cost of solving the large-scale Lyapunov equation (3.3) at each iteration. In section 3, we discussed the Krylov method which searches for an approximate solution $\mathbf{V}Q\mathbf{V}^T$, where $\mathbf{V}$ is an orthonormal basis of the Krylov subspace (3.6) and Q solves the small projected problem obtained by imposing the Galerkin condition. As shown in Example 1 (driven-cavity flow), a large Krylov subspace is needed for this method to compute an accurate enough solution even for the mild tolerance $\|R_\ell^{lyap}\|_F <$

$\|R_\ell^{eig}\|_F$. This deficiency leads us to the exploration of alternative Lyapunov solvers.

A recently developed projection method is the Rational Krylov Subspace Method (RKSM) [12]. Like the standard Krylov method, it projects a large Lyapunov equation onto a much smaller subspace, solves the small Lyapunov equation obtained by imposing the Galerkin condition and projects the solution back to the original space. In this method, the Krylov subspace is defined to be

$$\mathcal{K}_m(S, P, \mathbf{s}) = span\left\{P, (S - s_1 I)^{-1}P, \cdots, \prod_{j=1}^{m-1}(S - s_{m-j}I)^{-1}P\right\} \tag{3.17}$$

where $\mathbf{s} = [s_1, s_2, \cdots, s_{m-1}]^T \in \mathbb{C}^{m-1}$ is a vector of shifts that can be selected *a priori* or generated adaptively during computation. An algorithm that computes a decomposition similar to (3.7) for $\mathcal{K}_m(S, P, \mathbf{s})$ can be found in [37]. The use of such a subspace is first introduced by Ruhe for eigenvalue computation [36], where the shifts are placed around the target eigenvalues. In [11], RKSM is used to approximate $u(t) = \exp(St)u(0) \in \mathbb{R}^n$ where $S \in \mathbb{R}^{n\times n}$ is symmetric negative definite. An adaptive approach of choosing the shifts is proposed in [11] with the goal of minimizing the upper bound of the $L_2(0, \infty)$ error of the RKSM solution. This upper bound suggests that the shifts should lie on the imaginary axis, although it is shown in [11] that they can be restricted to the interval $[-\theta_{max}, -\theta_{min}]$ on the real line, where θ_{max} and θ_{min} are the largest and smallest eigenvalues of S, respectively. We present the formula for computing the next shift s_{m+1} $(m \geq 1)$ proposed in [11]

without going into detail:

$$s_{m+1} = \arg\left(\max_{s\in\mathcal{I}} \frac{1}{|r_m(s)|}\right),\ r_m(z) = \frac{\prod_{j=1}^{m}\left(z-\widehat{\theta}_j\right)}{\prod_{j=1}^{m}(z-s_j)} \tag{3.18}$$

where $\left\{\widehat{\theta}_j\right\}_{j=1}^{m}$ are the Ritz values of S on the Krylov subspace $\mathcal{K}_m = (S, P, \mathbf{s})$, $\{s_j\}_{j=1}^{m}$ are the shifts computed in previous iterations, and $\mathcal{I} = [-\theta_{max}, -\theta_{min}]$. In each Arnoldi step, a new pole will be added to the denominator of $r_m(z)$ and the numerator of $r_m(z)$ will be completely changed. To start the computation, the first shift s_1 is set to be an estimate of $-\theta_{max}$ or $-\theta_{min}$, which must be provided by some means. In [12], it is shown that this adaptive computation of the shifts can be used to generate an efficient Krylov subspace for solving the Lyapunov equation (3.3). This is motivated by the relation between $\exp(St)P$ and the analytic solution $-\int_0^\infty \exp(St)PCP^T \exp(S^T t)\, dt$ to (3.3). To generate the adaptive approach to a nonsymmetric S, [12] suggests replacing $\mathcal{I} = [-\theta_{max}, -\theta_{min}]$ by $\mathcal{I} = [-Re_{max}(\theta), -Re_{min}(\theta)]$. As before, $Re_{max}(\theta)$ and $Re_{min}(\theta)$ must be estimated beforehand (see [12] for a discussion). A convergence analysis of RKSM is given in [10].

Recall that in our problem, $S = \mathbf{A}^{-1}\mathbf{M}$ where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is the Jacobian matrix and $\mathbf{M} \in \mathbb{R}^{n\times n}$ is the nonsingular, shifted mass matrix given by (3.16). Both $\mathbf{A}$ and $\mathbf{M}$ are large and sparse and $\mathbf{A}$ is nonsymmetric. Each iteration of the standard Krylov method applied to (3.3) requires p solves with the coefficient matrix $\mathbf{A}$ to compute the new Arnoldi block V_{m+1}, where p is rank of the right-hand side of (3.3). On the other hand, each iteration of RKSM requires p solves with

the coefficient matrices $\mathbf{M} - s_j\mathbf{A}$ to compute the new Arnoldi block V_{m+1}, and an extra p solves with the coefficient matrix $\mathbf{A}$ to compute SV_{m+1}, which in turn gives us the Rayleigh quotient $\mathbf{V}^T S\mathbf{V}$ (see Proposition 4.1 from [12]). While we can pre-factor $\mathbf{A}$ or pre-compute a preconditioner for $\mathbf{A}$, we cannot do the same thing for $\mathbf{M} - s_j\mathbf{A}$, since the shift s_j is different from iteration to iteration. Therefore, when Krylov subspaces of the same dimension are used to approximate the solution to (3.3), RKSM will be more expensive than the standard Krylov method. RKSM is only competitive when it can generate the solution with a smaller subspace.

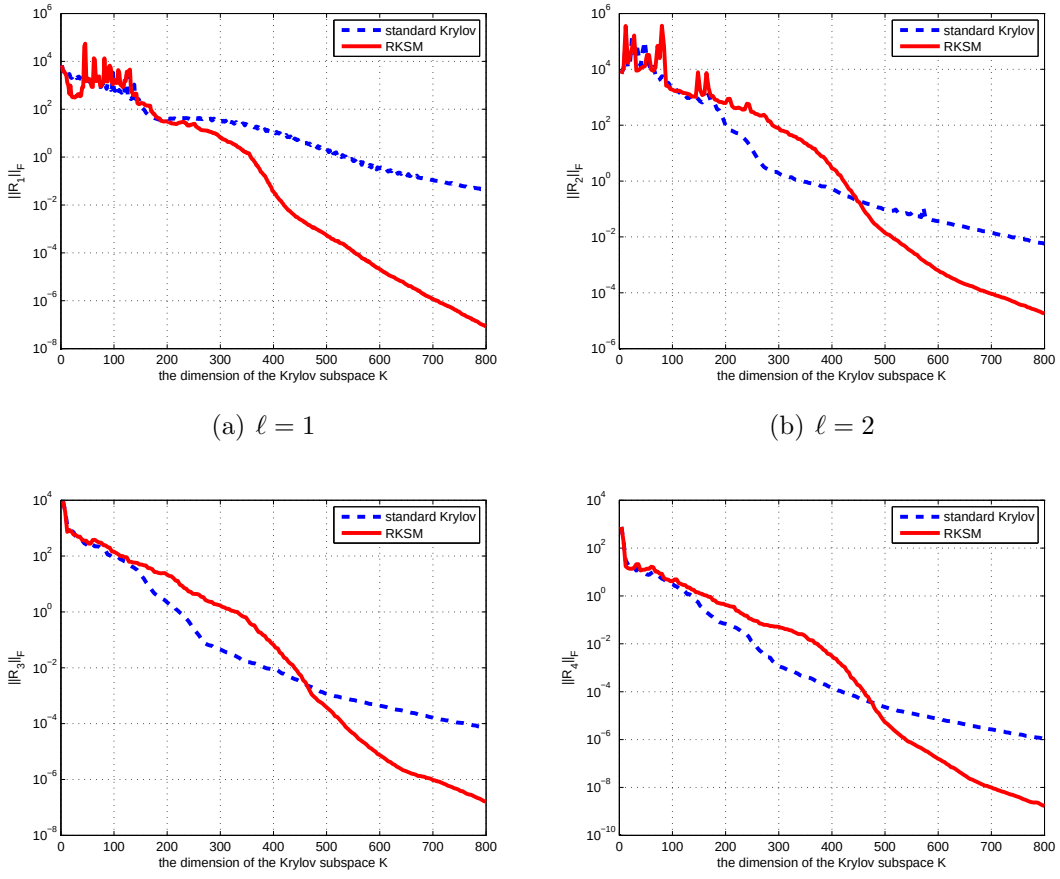
The examples we consider in this discussion of Lyapunov solvers are the Lyapunov equations

$$SY_\ell + Y_\ell S^T = P_\ell C_\ell P_\ell^T, \ \ell = 1, 2, 3, 4 \tag{3.19}$$

arising from the first three iterations of Algorithm 7 (with $\delta = 1$ and the standard Krylov Lyapunov solver) for driven-cavity flow on the two coarser meshes. The rank of the right-hand side is 2 for $\ell = 1$ and 4 for $\ell = 2, 3, 4$. The matrix $\mathbf{A}$ is pre-factored. Let the residual of (3.19) be R_ℓ^{lyap} as before. To compare the performance of the standard Krylov method and RKSM for solving (3.19), we have carried out the following numerical experiments.

We first compare the performance of the two methods. In Figure 3.4, we plot the decay of residual norm ($\|R_\ell^{lyap}\|_F$) for both the standard Krylov method and RKSM as the dimension of the Krylov subspace $\mathcal{K}$ (see (3.6) and (3.17) for definition) increases to an allowed maximum of 800. In all four cases, RKSM has a faster asymptotic convergence rate than the standard Krylov method, and RKSM

is able to find a much more accurate solution. For example, when $\ell = 1$, the final residual norm of the RKSM solution is about 10^{-7} whereas that of the Krylov solution is only about 10^{-1}.



(a) $\ell = 1$ (b) $\ell = 2$

(c) $\ell = 3$ (d) $\ell = 4$

Figure 3.4: Comparison of the standard Krylov method and RKSM for solving 3.19 in driven-cavity flow (64×64 mesh)

Secondly, we compare the CPU times of the standard Krylov method and RKSM as well as the ranks of the solutions produced by them, when the stopping criterion is $\|R_\ell^{lyap}\|_F < 10^{-3}$. These results are reported in Table 3.4. Consider first the case $\ell = 1$ where we performed the computation for two different mesh sizes. On both meshes, RKSM yields solutions with much lower rank, on the order

of 24% to 35% of the rank of the Krylov solutions. It is also much cheaper than the standard Krylov method in terms of CPU time; for example, on the coarsest mesh, it takes RKSM 8 minutes to compute the solution with rank 426 but 63 minutes for the standard Krylov method to compute the solution of rank 1218. The high cost of RKSM *per* iteration is fully compensated for by its early convergence. In addition, when the mesh is refined, the rank of the RKSM solution seems to be mesh-independent (426 and 428) whereas the rank of the standard Krylov method increases noticeably (1218 and 1748). This suggests that the finer the mesh is, the more efficient RKSM will be compared with the Krylov method.

Next, consider analogous results for $\ell = 2, 3, 4$, *i.e.*, as the outer iteration proceeds. It can be seen from Figure 3.4 that for both Lyapunov solvers the Lyapunov equation becomes progressively easier to solve as ℓ increases. This is due to the fact that as ℓ increases, the starting block of both methods, P_ℓ, gets 'closer' to the eigenvectors associated with the dominant eigenvalues of the solution Y_ℓ, and moreover, these dominant eigenvalues become more separate from the other eigenvalues of Y_ℓ. Evidence for this is as follows. From section 2, we know that (normalized) Y_ℓ converges to the eigenvector $Z_c = \mathcal{V}\mathcal{D}\mathcal{V}^T$ of (2.16), where $\mathcal{V} \in \mathbb{R}^{n\times 2}$. Figure 3.5 shows the moduli of the 50 eigenvalues of Y_ℓ ($\ell = 1, 2, 3, 4$) with largest modulus. (The 800-dimensional RKSM solutions are taken as the exact Y_ℓ's.) It is not difficult to see that as ℓ increases, the two eigenvalues of Y_ℓ with largest modulus become more dominant. Let $U_\ell \in \mathbb{R}^{n\times 2}$ hold the eigenvectors of Y_ℓ associated with the two dominant eigenvalues, and let $\angle(P_\ell, U_\ell)$ denote the angle between the subspaces spanned by P_ℓ and U_ℓ (see [50] for definition of the angle between two subspaces). The

smaller this angle is, the closer the two subspaces are to being linearly dependent. We compute $\angle(P_\ell, U_\ell)$ for $\ell = 1, 2, 3, 4$ using Matlab function 'subspace' and obtain the following results: $\angle(P_1, U_1) \approx 1.5326$, $\angle(P_2, U_2) \approx 0.3564$, $\angle(P_3, U_3) \approx 0.0071$ and $\angle(P_4, U_4) \approx 0.0002$. Thus, $\angle(P_\ell, U_\ell)$ goes to zero rapidly as the outer iteration proceeds. Table 3.4 also suggests that the standard Krylov method becomes more

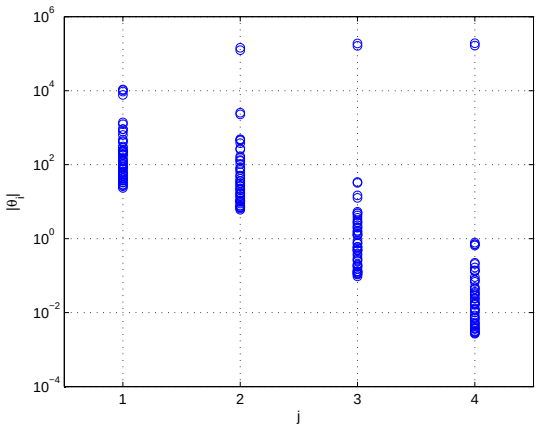
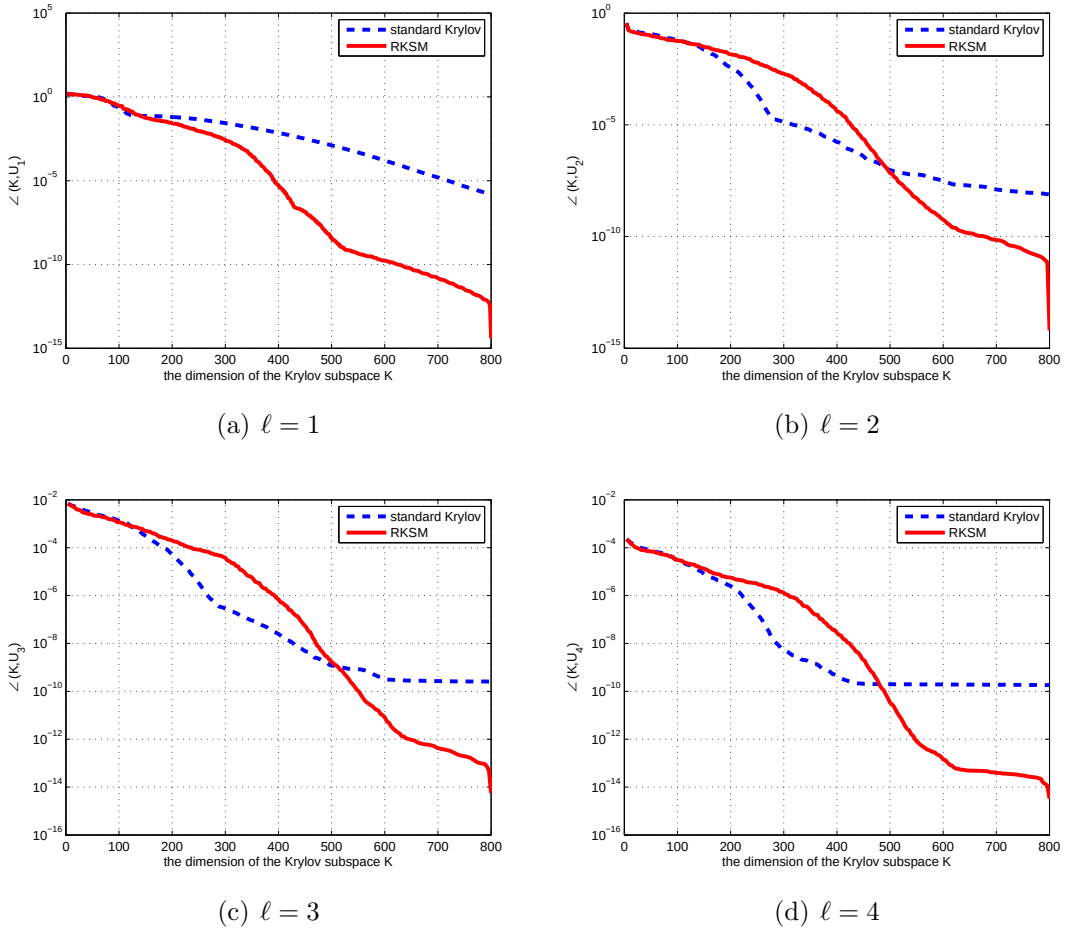


Figure 3.5: Moduli of the 50 eigenvalues of Y_ℓ with largest modulus (driven-cavity flow, 64×64 mesh)

advantageous compared to RKSM as the outer iteration proceeds. The reason behind is that the standard Krylov method is able to resolve the dominant eigenvectors U_ℓ of the solution Y_ℓ faster than RKSM in the case where $\angle(P_\ell, U_\ell)$ is already small. This point is demonstrated in Figure 3.6, which shows the decay of $\angle(\mathcal{K}, U_\ell)$, the angle between the subspace spanned by U_ℓ and the Krylov subspace. The decay curves in Figure 3.4 and Figure 3.6 are clearly similar.

We conclude section 5.3 with the following remarks.

1. If the goal is to solve (3.3) accurately, RKSM is definitely the superior choice. Compared to the standard Krylov method, it has a faster asymptotic rate of



(a) $\ell = 1$ (b) $\ell = 2$

(c) $\ell = 3$ (d) $\ell = 4$

Figure 3.6: Decay of $\angle(\mathcal{K}, U_\ell)$ of the standard Krylov method and RKSM (driven-cavity flow, 64×64 mesh)

convergence, which leads to significant savings in both storage and CPU time; moreover, the rank of the solution it computes is mesh-independent, which is important for problems arising from discretization of 2- or 3-dimensional PDEs.

2. However, as pointed out in section 4, solving the Lyapunov equations accurately is not of primary interest in the current study, since we only need to solve it accurately enough for the outer iteration, *i.e.*, $\|R_\ell^{lyap}\|_F < \|R_\ell^{eig}\|_F$ (see Algorithm 7, step 2.4 with $\delta = 1$). In our experiments, $\|R_1^{eig}\|_F \approx 3.81 \times 10^2$,

$\|R_2^{eig}\|_F \approx 1.52 \times 10^{-1}$, $\|R_3^{eig}\|_F \approx 2.35 \times 10^{-3}$, and $\|R_4^{eig}\|_F \approx 7.17 \times 10^{-5}$ (see Table A.1 in the Appendix). Figure 3.4 shows that if the stopping criterion is this mild, the two methods require Krylov subspaces of almost the same dimension and therefore, RKSM will be the less effective choice between the two.

3.4 Conclusions

We have refined the Lyapunov inverse iteration proposed in [29] and examined the application of our algorithm to two examples arising from models of incompressible flow. The driven-cavity flow example is a particularly difficult problem. For both examples, the new algorithm is able to compute good estimates of the critical parameter value at which Hopf bifurcation takes place. Our algorithm belongs to the class of inner-outer iterative methods: the outer iteration is the inverse iteration for a special eigenvalue problem and the inner iteration is to solve a Lyapunov equation. Based on existing theory of inner-outer iterative methods, the Lyapunov equations do not need to be solved to high accuracy; instead, a mild tolerance is sufficient. In this scenario, the standard Krylov method is more effective than the rational Krylov subspace method for solving the large-scale Lyapunov systems that arise for our application. The method developed in this chapter has also been applied successfully in aeroelastic stability analysis (see [47]).

Table 3.3: Algorithm 7 applied to flow over an obstacle (128×512 mesh, $Re_0 = 320$) for $\delta = 1,\ 10^{-1},\ 10^{-2}$ in (3.15)

ℓ	$Re^{(\ell)}$	$\mu^{(\ell)}$	$\|r_\ell\|_2$	$\|R_\ell^{eig}\|_F$	$\|R_\ell^{lyap}\|_F$	d_ℓ	k_ℓ
			$\delta = 1$				
1	-331	-6.21192e-13	1.52776e-01	4.38125e+0	3.52097e+0	68	10
2	311	2.26820i	2.28878e-04	1.79583e-1	1.77367e-1	56	20
3	378	2.27689i	4.11801e-05	5.72823e-3	4.23584e-3	68	20
4	375	2.26633i	9.82413e-06	1.20449e-3	6.69375e-4	68	20
5	373	2.26632i	1.79954e-06	2.34683e-4	2.09699e-4	64	30
6	373	2.26661i	2.42540e-07	2.87217e-5	2.67124e-5	64	20
7	373	2.26656i	4.05258e-08	5.38433e-6	4.10305e-6	64	20
8	373	2.26656i	5.45124e-09	7.12393e-7	4.27719e-7	68	20
9	373	2.26656i	1.32615e-09	1.82166e-7	9.15168e-8	68	20
10	373	2.26656i	3.22020e-10	3.99332e-8	—	—	—
					Total:	588	
			$\delta = 10^{-1}$				
1	-331	-6.21192e-13	1.52776e-01	4.38125e+0	4.22028e-1	202	30
2	366	2.21977i	1.44453e-04	3.67019e-2	3.34052e-3	84	40
3	368	2.26650i	2.91683e-05	3.77305e-3	2.59429e-4	80	30
4	374	2.26727i	3.07688e-06	5.16995e-4	3.16904e-5	80	30
5	373	2.26650i	4.51259e-07	5.51444e-5	5.21710e-6	76	40
6	373	2.26657i	4.14640e-08	5.33721e-6	4.04173e-7	76	40
7	373	2.26656i	4.67350e-09	6.55972e-7	4.42397e-8	80	40
8	373	2.26656i	6.61702e-10	9.81259e-8	—	—	—
					Total:	678	
			$\delta = 10^{-2}$				
1	-331	-6.21192e-13	1.52776e-01	4.38125e+0	3.87099e-2	254	40
2	364	2.22687i	5.37359e-04	1.51434e-2	1.36842e-4	164	60
3	370	2.26840i	1.79202e-05	2.39139e-3	2.32630e-5	156	60
4	374	2.26678i	2.40915e-06	3.15403e-4	2.86933e-6	148	50
5	373	2.26653i	4.22143e-07	5.32762e-5	5.14646e-7	132	50
6	373	2.26657i	9.53373e-09	2.13741e-6	1.82336e-8	160	50
7	373	2.26656i	3.55204e-09	6.02453e-7	4.99433e-9	160	50
8	373	2.26656i	3.66251e-10	5.45473e-8	—	—	—
					Total:	1174	

Table 3.4: Rank of the approximate solution and CPU time

	64×64 mesh				128×128 mesh			
ℓ	Krylov		RKSM		Krylov		RKSM	
1	1218	(63 min.)	426	(8 min.)	1748	(276 min.)	428	(34 min.)
2	1024	(15 min.)	588	(10 min.)				
3	516	(1 min.)	476	(6 min.)				
4	312	(0.3 min.)	428	(5 min.)				

Chapter 4

Lyapunov Inverse Iteration for Computing a Few Rightmost Eigenvalues of Large Generalized Eigenvalue Problems

The method proposed in [29] and refined in Chapter 3 allows us to estimate the critical parameter value α_c at which stability is lost without computing the rightmost eigenvalue of (2.2). However, it only works in the neighborhood of the critical point $(\overline{u}_c, \alpha_c)$. In Chapter 3, for instance, the critical point of all numerical examples is known *a priori*, so that we can pick a point $(\overline{u}_0, \alpha_0)$ close to it and apply Lyapunov inverse iteration with confidence. Usually α_c is unknown and we start from a point $(\overline{u}_0, \alpha_0)$ in the stable regime of the solution path $\mathcal{S}$ that may be distant from the critical point. In this scenario, the method of [29] and Chapter 3 cannot be used to estimate α_c, since the Jacobian matrix $\mathcal{J}(\alpha_c)$ at the critical point cannot be approximated by $\mathbf{A} + \lambda_c \mathbf{B}$, where $\mathbf{A} = \mathcal{J}(\alpha_0)$, $\mathbf{B} = \frac{d\mathcal{J}}{d\alpha}(\alpha_c)$ and $\lambda_c = \alpha_c - \alpha_0$. However, quantitative information about how far away $(\overline{u}_0, \alpha_0)$ is from $(\overline{u}_c, \alpha_c)$ can still be obtained by estimating the distance between the rightmost eigenvalue of (2.2) at α_0 and the imaginary axis: if the rightmost eigenvalue is far away from the imaginary axis, then it is reasonable to assume that $(\overline{u}_0, \alpha_0)$ is far away from the critical point as well, and therefore we should march along $\mathcal{S}$ using numerical continuation until we are close enough to $(\overline{u}_c, \alpha_c)$; otherwise, we can assume that $(\overline{u}, \alpha_0)$ is already in the neighborhood of the critical point and the method of [29]

and Chapter 3 can be applied to estimate α_c.

The goal of this chapter is to develop a robust method to compute a few rightmost eigenvalues of (2.2) in the stable regime of $\mathcal{S}$. The plan of this chapter is as follows. In section 4.1, we show that the distance between the imaginary axis and the rightmost eigenvalue of (2.2) is the eigenvalue with smallest modulus of an eigenvalue problem similar in structure to (3.14). As a result, this eigenvalue can also be computed efficiently by Lyapunov inverse iteration. In section 4.2, we present numerical results for several examples arising from fluid dynamics, and provide an analysis of the fast convergence of Lyapunov inverse iteration observed in our experiments. Based on the analysis, we also propose a modified algorithm that guarantees convergence in only two iterations. In section 4.3, we show that the analysis in sections 4.1 and 4.2 can be generalized to a deflated version of the Lyapunov eigenvalue problem, which leads to an algorithm for computing k ($1 \leq k \ll n$) rightmost eigenvalues of (2.2). Details of the implementation of this algorithm are discussed in section 5. Finally, we make some concluding remarks in section 4.4.

4.1 Computing the Distance between the Rightmost Eigenvalue and the Imaginary Axis

Let $(\overline{u}_0, \alpha_0)$ be any point in the stable regime of $\mathcal{S}$ and assume $\mathbf{M}$ is nonsingular in (2.2). Let (μ_j, x_j) ($\|x_j\|_2 = 1$, $j = 1, 2, \ldots, n$) be the eigenpairs of (2.2) at α_0, where the real parts of μ_j, $Re(\mu_j)$, are in decreasing order, *i.e.*, $0 > Re(\mu_1) \geq Re(\mu_2) \geq \ldots \geq Re(\mu_n)$. Then the distance between the rightmost eigenvalue and

the imaginary axis is $-Re(\mu_1)$. Let $\mathbf{A} = \mathcal{J}(\alpha_0)$ and $S = \mathbf{A}^{-1}\mathbf{M}$. To compute this distance, we first observe that $-Re(\mu_1)$ is the eigenvalue with smallest modulus of the $n^2 \times n^2$ generalized eigenvalue problem

$$(\Psi_1 + \lambda\Psi_0)\, z = 0 \tag{4.1}$$

where $\Psi_1 = S \otimes I + I \otimes S$ and $\Psi_0 = 2S \otimes S$ (I is the identity matrix of order n). We proceed in two steps to prove this assertion. First, we show that $-Re(\mu_1)$ is an eigenvalue of (4.1).

Theorem 4.1. *Assume* $\mathbf{M}$ *is nonsingular. The eigenvalues of* (4.1) *are* $\lambda_{i,j} = -\frac{1}{2}(\mu_i + \mu_j)$, $i, j = 1, 2, \ldots, n$. *For any pair* (i, j), *there are eigenvectors associated with* $\lambda_{i,j}$ *given by* $z_{i,j} = x_i \otimes x_j$ *and* $z_{j,i} = x_j \otimes x_i$.

Proof. Since (μ_j, x_j) $(j = 1, 2, \ldots, n)$ are the eigenpairs of $\mathbf{A}x = \mu\mathbf{M}x$, $\left(\frac{1}{\mu_j}, x_j\right)$ are the eigenpairs of S. We first prove that the eigenvalues of (4.1) are $\{\lambda_{i,j}\}_{i,j=1}^{n}$. Let J be the Jordan normal form of S and P be an invertible matrix such that $S = PJP^{-1}$. Then

$$\begin{aligned}
(-\Psi_0)^{-1}\,\Psi_1(P \otimes P) &= -\frac{1}{2}(PJP^{-1} \otimes PJP^{-1})^{-1}(PJP^{-1} \otimes I + I \otimes PJP^{-1})(P \otimes P) \\
&= -\frac{1}{2}\left(PJ^{-1}P^{-1} \otimes PJ^{-1}P^{-1}\right)(PJ \otimes P + P \otimes PJ) \\
&= -\frac{1}{2}(P \otimes P)\left(J^{-1}P^{-1} \otimes J^{-1}P^{-1}\right)(PJ \otimes P + P \otimes PJ) \\
&= (P \otimes P)\left[-\frac{1}{2}\left(I \otimes J^{-1} + J^{-1} \otimes I\right)\right].
\end{aligned}$$

This implies that (4.1) and $-\frac{1}{2}\left(I \otimes J^{-1} + J^{-1} \otimes I\right)$ have the same eigenvalues. Due to the special structure of the Jordan normal form J, $I \otimes J^{-1} + J^{-1} \otimes I$ is an upper triangular matrix whose diagonal entries are $\{\mu_i + \mu_j\}_{i,j=1}^n$. Consequently, the eigenvalues of $-\frac{1}{2}\left(I \otimes J^{-1} + J^{-1} \otimes I\right)$ are $\left\{-\frac{1}{2}(\mu_i + \mu_j)\right\}_{i,j=1}^n = \{\lambda_{i,j}\}_{i,j=1}^n$. Therefore, the eigenvalues of (4.1) are $\{\lambda_{i,j}\}_{i,j=1}^n$ as well.

Second, we show that $z_{i,j}$ is an eigenvector of (4.1) associated with the eigenvalue $\lambda_{i,j}$. For any pair (i,j) $(i,j = 1,2,\ldots,n)$,

$$\Psi_1(x_i \otimes x_j) = (S \otimes I + I \otimes S)(x_i \otimes x_j) = (Sx_i) \otimes x_j + x_i \otimes (Sx_j) = \left(\frac{1}{\mu_i} + \frac{1}{\mu_j}\right)(x_i \otimes x_j),$$

and

$$\Psi_0(x_i \otimes x_j) = 2(S \otimes S)(x_i \otimes x_j) = 2(Sx_i) \otimes (Sx_j) = \frac{2}{\mu_i \mu_j}(x_i \otimes x_j).$$

Therefore, $\Psi_1 z_{i,j} = \left(\frac{1}{\mu_i} + \frac{1}{\mu_j}\right)\frac{\mu_i\mu_j}{2}\Psi_0 z_{i,j} = \lambda_{i,j}\left(-\Psi_0\right) z_{i,j}$. Similarly, we can show that $\Psi_1 z_{j,i} = \lambda_{i,j}\left(-\Psi_0\right) z_{j,i}$. □

If μ_1 is real, then $-Re(\mu_1) = -\mu_1 = -\frac{1}{2}(\mu_1 + \mu_1) = \lambda_{1,1}$; if μ_1 is not real (*i.e.*, $\mu_1 = \overline{\mu_2}$ and $x_1 = \overline{x_2}$), then $-Re(\mu_1) = -\frac{1}{2}(\mu_1 + \overline{\mu_1}) = -\frac{1}{2}(\mu_1 + \mu_2) = \lambda_{1,2} = \lambda_{2,1}$. In both cases, by Theorem 4.1, $-Re(\mu_1)$ is an eigenvalue of (4.1).

We next show that $-Re(\mu_1)$ is the eigenvalue of (4.1) with smallest modulus.

Theorem 4.2. *Assume all the eigenvalues of* $\mathbf{A}x = \mu\mathbf{M}x$ *lie in the left half of the complex plane. Then the eigenvalue of* (4.1) *with smallest modulus is* $-Re(\mu_1)$.

Proof. Let $\mu_j = a_j + ib_j$. Then $0 > a_1 \geq a_2 \geq \ldots \geq a_n$. If the rightmost

eigenvalue of $\mathbf{A}x = \mu\mathbf{M}x$ is real, then $-Re(\mu_1) = \lambda_{1,1}$, and since $0 > a_1 \geq a_2 \geq \ldots \geq a_n$, it follows that

$$|\lambda_{1,1}|^2 = \frac{1}{4}(a_1 + a_1)^2 \leq \frac{1}{4}\left[(a_i + a_j)^2 + (b_i + b_j)^2\right] = |\lambda_{i,j}|^2$$

for any pair (i, j). Alternatively, if the rightmost eigenvalues of $\mathbf{A}x = \mu\mathbf{M}x$ consist of a complex conjugate pair, then $a_1 = a_2$, $b_1 = -b_2$, $-Re(\mu_1) = \lambda_{1,2} = \lambda_{2,1}$, and similarly,

$$|\lambda_{1,2}|^2 = |\lambda_{2,1}|^2 = \frac{1}{4}\left[(a_1 + a_1)^2 + (b_1 - b_1)^2\right] \leq \frac{1}{4}\left[(a_i + a_j)^2 + (b_i + b_j)^2\right] = |\lambda_{i,j}|^2$$

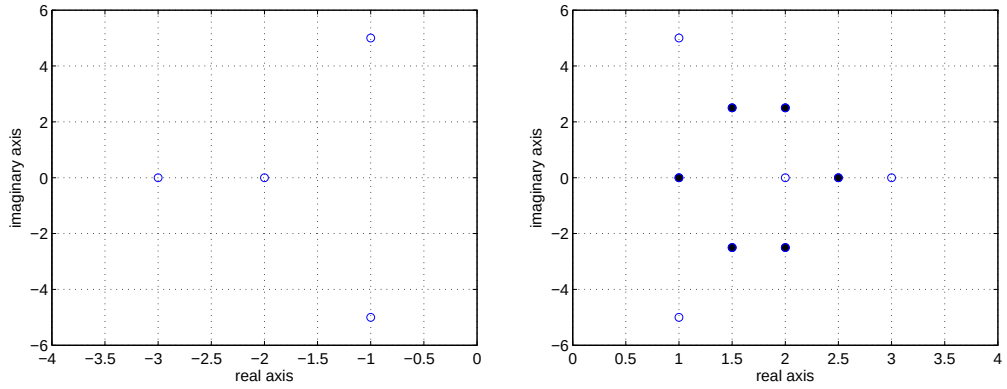
for any pair (i, j). In both cases, $-Re(\mu_1)$ is the eigenvalue of (4.1) with smallest modulus. □

Simple example: Consider a 4×4 example of $\mathbf{A}x = \mu\mathbf{M}x$ whose eigenvalues are $\mu_{1,2} = -1 \pm 5i$, $\mu_3 = -2$ and $\mu_4 = -3$ (see Figure 4.1(a)). The eigenvalues of the corresponding 16×16 eigenvalue problem (4.1) are plotted in Figure 4.1(b) and listed in Table 4.1. As seen in Figure 4.1(b), $\lambda_{1,1} = 1 - 5i$, $\lambda_{1,2} = \lambda_{2,1} = 1$ and $\lambda_{2,2} = 1 + 5i$ are the leftmost eigenvalues of (4.1), and $\lambda_{1,2} = \lambda_{2,1}$ are the eigenvalues of (4.1) with smallest modulus.

Table 4.1: The eigenvalues of (4.1) corresponding to the 4×4 example

$\lambda_{1,1} = 1 - 5i$			
$\lambda_{2,1} = \lambda_{1,2} = 1$	$\lambda_{2,2} = 1 + 5i$		
$\lambda_{3,1} = \lambda_{1,3} = 1.5 - 2.5i$	$\lambda_{3,2} = \lambda_{2,3} = 1.5 + 2.5i$	$\lambda_{3,3} = 2$	
$\lambda_{4,1} = \lambda_{1,4} = 2 - 2.5i$	$\lambda_{4,2} = \lambda_{2,4} = 2 + 2.5i$	$\lambda_{4,3} = \lambda_{3,4} = 2.5$	$\lambda_{4,4} = 3$

Assume $\mathbf{A}x = \mu\mathbf{M}x$ has a complete set of eigenvectors $\{x_j\}_{j=1}^n$. Then (4.1)



(a) The spectrum of $\mathbf{A}x = \mu\mathbf{M}x$ (b) The spectrum of (4.1)

Figure 4.1: The spectrum of $\mathbf{A}x = \mu\mathbf{M}x$ and (4.1) for the 4×4 example ($\bullet$: double eigenvalues).

also has a complete set of eigenvectors $\{z_{i,j}\}_{i,j=1}^{n}$. By Theorem 4.2, the distance between the imaginary axis and the rightmost eigenvalue, $-Re(\mu_1)$, can be found by inverse iteration applied to (4.1), which involves solving linear systems of order n^2. In [29] and Chapter 3, an $n^2 \times n^2$ eigenvalue problem (2.15) similar in structure to (4.1) is dealt with by rewriting an equation of Kronecker sums into an equation of Lyapunov form, *i.e.*, (2.16). Here, similarly, we can rewrite (4.1) into

$$SZ + ZS^T + \lambda(2SZS^T) = 0. \tag{4.2}$$

Any eigenpair (λ, z) of (4.1) is related to an eigenpair (λ, Z) of (4.2) by $z = vec(Z)$. By Theorem 4.1 and the relation between (4.1) and (4.2), $(\lambda_{i,j}, Z_{i,j})$ $(i, j = 1, 2, \ldots, n)$ are the eigenpairs of (4.2) where $Z_{i,j} = x_j x_i^T$; in addition, by Theorem 4.2, $-Re(\mu_1)$ is the eigenvalue of (4.2) with smallest modulus. Furthermore, under certain conditions, $-Re(\mu_1)$ is an eigenvalue of (4.2) whose associated eigenvector is real, symmetric and of low rank. Assume the following:

(a1) for any $1 < i \leq n$, if $Re(\mu_i) = Re(\mu_1)$, then $\mu_i = \overline{\mu_1}$; and

(a2) μ_1 is a simple eigenvalue of $\mathbf{A}x = \mu\mathbf{M}x$.

Consequently, if μ_1 is real, then $-Re(\mu_1)$ is a simple eigenvalue of (4.1) with the eigenvector $z_{1,1} = x_1 \otimes x_1$; otherwise, $-Re(\mu_1)$ is a double eigenvalue of (4.1) with the eigenvectors $z_{1,2} = x_1 \otimes \overline{x_1}$ and $z_{2,1} = \overline{x_1} \otimes x_1$. When the eigenvectors of (4.2) are restricted to the subspace of $\mathbb{C}^{n\times n}$ consisting of symmetric matrices Z, then by Theorem 2.3 from [29], $-Re(\mu_1)$ has a unique (up to a scalar multiplier), real and symmetric eigenvector $x_1x_1^T$ (if μ_1 is real), or $x_1x_1^* + \overline{x_1}x_1^T$ (if μ_1 is not real) where x_1^* denotes the conjugate transpose of x_1. Therefore, we can apply Lyapunov inverse iteration (see Algorithm 4) to (4.2) to find $-Re(\mu_1)$, the eigenvalue of (4.2) with smallest modulus:

Algorithm 8: Lyapunov inverse iteration for (4.2)

---

1. Given $\mathbf{V}_1 \in \mathbb{R}^n$ with $\|\mathbf{V}_1\|_2 = 1$.
2. For $\ell = 1, 2, \cdots$
   - 2.1. Rank reduction[1]: compute $\widetilde{S} = \mathbf{V}_\ell^T S \mathbf{V}_\ell$ and solve for the eigenvalue $\widetilde{\lambda}_1$ of
   $$\widetilde{S}\widetilde{Z} + \widetilde{Z}\widetilde{S}^T + \widetilde{\lambda}\left(2\widetilde{S}\widetilde{Z}\widetilde{S}^T\right) = 0 \tag{4.3}$$
   with smallest modulus and its eigenvector $\widetilde{Z}_1 = \widetilde{\mathcal{V}}\widetilde{\mathcal{D}}\widetilde{\mathcal{V}}^T$.
   - 2.2. Set $\lambda^{(\ell)} = \widetilde{\lambda}_1$ and $Z^{(\ell)} = \mathcal{V}_\ell\widetilde{\mathcal{D}}\mathcal{V}_\ell^T$, where $\mathcal{V}_\ell = \mathbf{V}_\ell\widetilde{\mathcal{V}}$.
   - 2.3. If $\left(\lambda^{(\ell)}, Z^{(\ell)}\right)$ is accurate enough, then stop.
   - 2.4. Else, solve for Y_ℓ from
   $$SY_\ell + Y_\ell S^T = -2SZ^{(\ell)}S^T \tag{4.4}$$
   in factored form: $Y_\ell = \mathbf{V}_{\ell+1}D_{\ell+1}\mathbf{V}_{\ell+1}^T$.

As the iteration proceeds, the iterate $\left(\lambda^{(\ell)}, Z^{(\ell)} = \mathcal{V}_\ell\widetilde{D}\mathcal{V}_\ell^T\right)$ will converge to $(-Re(\mu_1), \mathcal{V}\mathcal{D}\mathcal{V}^T)$ where $\|\mathcal{D}\|_F = 1$ and $\mathcal{V} = x_1$ (if μ_1 is real) or $\mathcal{V} \in \mathbb{R}^{n\times 2}$ is

[1]When $\ell = 1$, (4.3) is a scalar equation and its eigenpair is $\left(\widetilde{\lambda}_1, \widetilde{Z}_1\right) = \left(-\widetilde{S}^{-1}, 1\right)$.

an orthonormal matrix whose columns span $\{x_1, \overline{x_1}\}$ (if μ_1 is not real). Besides estimates of $-Re(\mu_1)$, we can also obtain from Algorithm 8 estimates of (μ_1, x_1) by solving the small $q \times q$ ($q = 1$ or 2) eigenvalue problem

$$\left(\mathcal{V}_\ell^T S \mathcal{V}_\ell\right) y = \theta y \tag{4.5}$$

and taking $\mu^{(\ell)} = \frac{1}{\theta}$ and $x^{(\ell)} = \mathcal{V}_\ell y$. As $\mathcal{V}_\ell$ converges to $\mathcal{V}$, $\left(\mu^{(\ell)}, x^{(\ell)}\right)$ will converge to (μ_1, x_1).

At each iteration of Algorithm 8, a large-scale Lyapunov equation (4.4) needs to be solved. As has been done for (2.21), we can rewrite (4.4) as (2.24) (see Chapter 3 for details) where P_ℓ is orthonormal and is of rank 1 ($\ell = 1$) or 2 ($\ell > 1$). The Lyapunov equation (4.4) can then be solved by applying an iterative Lyapunov solver such as the "standard" Krylov method (see Algorithm 5) to (2.24). In step 2.1 (rank reduction) of Algorithm 8, although it may look like computing the projection $\widetilde{S} = \mathbf{V}_\ell^T S \mathbf{V}_\ell$ requires extra linear solves with coefficient matrix $\mathbf{A}$, in fact, if a Krylov-type method is used to solve (2.24), $\widetilde{S}$ can be obtained from the Arnoldi decomposition computed by the Krylov-type method for no additional cost when $\ell > 1$. For example, if the standard Krylov method is used to solve (2.24), by (3.7), $\widetilde{S} = H_m$ where m is the number of block Arnoldi steps needed.

4.2 Numerical Results

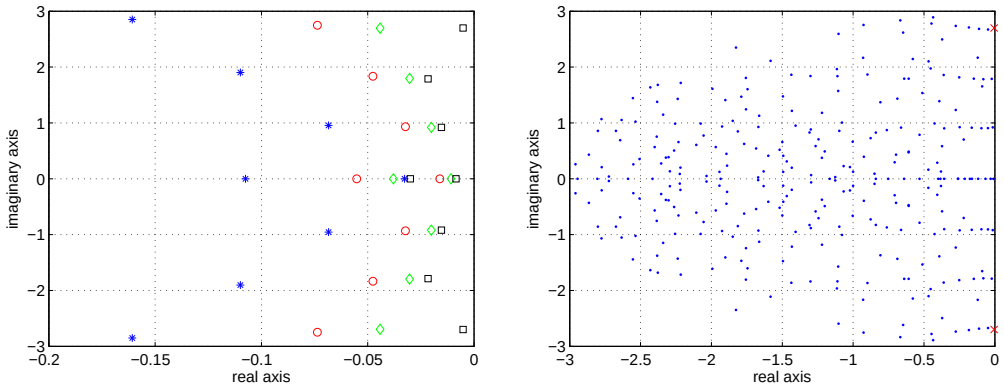
In this section, we test Algorithm 8 on several problems arising from fluid dynamics. As already noted in section 3.3, when (2.1) comes from a standard (*e.g.*, finite ele-

ment) discretization of the incompressible Navier-Stokes equations, the mass matrix $\mathbf{M}$ is singular, leading to infinite eigenvalues of (2.2) and singular $S = \mathbf{A}^{-1}\mathbf{M}$. We therefore used the shifted, nonsingular mass matrix (3.16) proposed in [8], which maps the infinite eigenvalues of (2.2) to finite ones away from the imaginary axis and leaves the finite eigenvalues of (2.2) unchanged. As in section 3.3, here $\mathbf{M}$ refers to this shifted mass matrix.

4.2.1 Example 1: Driven-Cavity Flow

The Q_2-Q_1 mixed finite element discretization (with a 64×64 mesh) of the Navier-Stokes equations gives rise to a generalized eigenvalue problem (2.2) of order $n = 9539$. Figure 4.2(a) depicts the path traced out by the eight rightmost eigenvalues of (2.2) for Reynolds numbers $Re = 2000, 4000, 6000, 7800$, at which the steady-state solution to (2.1) is stable. As the Reynolds number increases, the following trend can be observed: the eight rightmost eigenvalues all move towards the imaginary axis, and they become more clustered as they approach the imaginary axis. In addition, although the rightmost eigenvalue starts off being real, one conjugate pair of complex eigenvalues (whose imaginary parts are about $\pm 3i$) move faster towards the imaginary axis than the other eigenvalues and eventually they become the rightmost. They first cross the imaginary axis at $Re \approx 7929$ (see Table A.1), causing instability in the steady-state solution of (2.1). Finding the conjugate pair of rightmost eigenvalues of (2.2) at a high Reynolds number (for example, at $Re = 7800$) using an iterative eigenvalue solver such as the IRA method can be difficult,

as noted in section 3.3.



(a) (b)

Figure 4.2: (a) The eight rightmost eigenvalues for driven-cavity flow at different Reynolds numbers ($*: Re = 2000$, $\circ: Re = 4000$, $\diamondsuit: Re = 6000$, $\square: Re = 7800$). (b) The 300 eigenvalues with smallest modulus at $Re = 7800$ ($\times$: the rightmost eigenvalues).

For four various Reynolds numbers between 2000 and 7800, we apply Algorithm 8 (with the Rational Krylov Subspace Method [12] (RKSM) as the Lyapunov solver) to calculate the distance between the rightmost eigenvalue of (2.2) and the imaginary axis. The results are reported in Table 4.3 (see Table 4.2 for notation). The initial guess $\mathbf{V}_1$ is chosen to be a random vector of unit norm in $\mathbb{R}^n$, the stopping

Table 4.2: Notation for Algorithm 8

Symbol	Definition
$\lambda^{(\ell)}$, $Z^{(\ell)}$	the estimated eigenvalue of (4.2) with smallest modulus ($-Re(\mu_1)$) and its associated eigenvector, respectively
$\mu^{(\ell)}$	the estimated rightmost eigenvalue of $\mathbf{A}x = \mu\mathbf{M}x$, *i.e.*, μ_1
R_ℓ^{eig}	$SZ^{(\ell)} + Z^{(\ell)}S^T + \lambda^{(\ell)}(2SZ^{(\ell)}S^T)$, *i.e.*, the residual of the (4.2)
Y_ℓ^{approx}	the approximate solution to (2.24)
R_ℓ^{lyap}	$SY_\ell^{approx} + Y_\ell^{approx}S^T - P_\ell C_\ell P_\ell^T$, *i.e.*, the residual of (2.24)
d_ℓ	rank of Y_ℓ^{approx}

criterion for the eigenvalue residual is

$$\|R_\ell^{eig}\|_F < 10^{-8}, \tag{4.6}$$

and the stopping criterion for the Lyapunov solve is

$$\|R_\ell^{lyap}\|_F < 10^{-9} \cdot \left\|P_\ell C_\ell P_\ell^T\right\|_F = 10^{-9} \cdot \|C_\ell\|_F \,. \tag{4.7}$$

Note that both residual norms $\|R_\ell^{eig}\|_F$ and $\|R_\ell^{lyap}\|_F$ are cheap to compute (see section 3.2 for details). Therefore, the main cost of each iteration is about d_ℓ linear solves with coefficient matrix $\mathbf{M} - s_j\mathbf{A}$ and about d_ℓ linear solves with coefficient matrix $\mathbf{A}$ (see section 3.3.3 for a discussion on the cost of RKSM). All linear systems are solved using direct methods. As shown in Table 4.3, the distances between the rightmost eigenvalue of (2.2) and the imaginary axis at $Re = 2000, 4000, 6000, 7800$ are 0.03264, 0.01608, 0.01084, 0.00514, respectively. We also obtain estimates of the rightmost eigenvalue of (2.2) at the four Reynolds numbers: -0.03264, -0.01608, -0.01084, and -0.00514+2.69845i.

We note two trends seen in these results. First, surprisingly, for all the Reynolds numbers considered, Algorithm 8 converges to the desired tolerance (4.6) in only 2 iterations. That is, only the first Lyapunov equation

$$SY_1 + Y_1S^T = P_1C_1P_1^T \tag{4.8}$$

needs to be solved, where $P_1 \in \mathbb{R}^n$ and $C_1 \in \mathbb{R}$. Second, as the Reynolds number

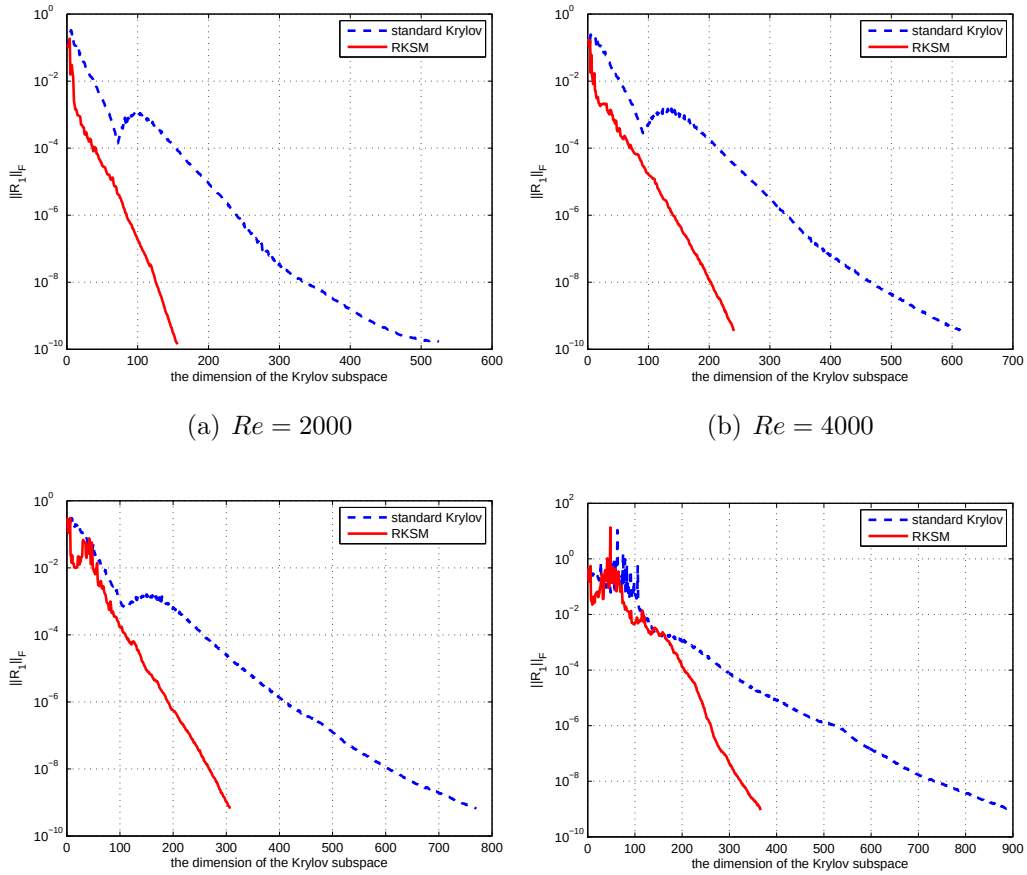
increases, it becomes more expensive to solve the Lyapunov equation to the same order of accuracy (4.7), since Krylov subspaces of increasing dimension are needed (156, 241, 307 and 366 for the four Reynolds numbers). We also tested Algorithm 8 using the standard Krylov method (Algorithm 5) to solve the Lyapunov systems. To solve (4.8) to the same accuracy, this method requires subspaces of dimension 525, 614, 770 and 896 for the four Reynolds numbers, which are more than twice as large as those required by RKSM (see Figure 4.3 for comparison). As a result, RKSM is much more efficient.

Table 4.3: Algorithm 8 applied to driven-cavity flow

ℓ	$\lambda^{(\ell)}$	$\mu^{(\ell)}$	$\|R_\ell^{eig}\|_F$	$\|R_\ell^{lyap}\|_F$	d_ℓ
			Re=2000		
1	884.383	-884.383	1.32049e+02	1.40794e-10	156
2	0.03264	-0.03264	2.56263e-11	—	—
			Re=4000		
1	-17765.8	17765.8	6.58651e+03	3.52618e-10	241
2	0.01608	-0.01608	4.25055e-10	—	—
			Re=6000		
1	1301.24	-1301.24	8.55652e+02	6.52387e-10	307
2	0.01084	-0.01084	7.11628e-10	—	—
			Re=7800		
1	695.951	-695.951	6.58622e+02	9.02875e-10	366
2	0.00514	-0.00514+2.69845i	3.62567e-11	—	—

4.2.2 Example 2: Flow over an Obstacle

The Q_2-Q_1 mixed finite element discretization (with a 32×128 mesh) of the Navier-Stokes equations gives rise to a generalized eigenvalue problem (2.2) of order $n = 9512$. Figure 4.4(a) depicts the path traced out by the six rightmost eigenvalues of (2.2) for $Re = 100, 200, 300, 350$ in the stable regime, and Figure 4.4(b) shows the



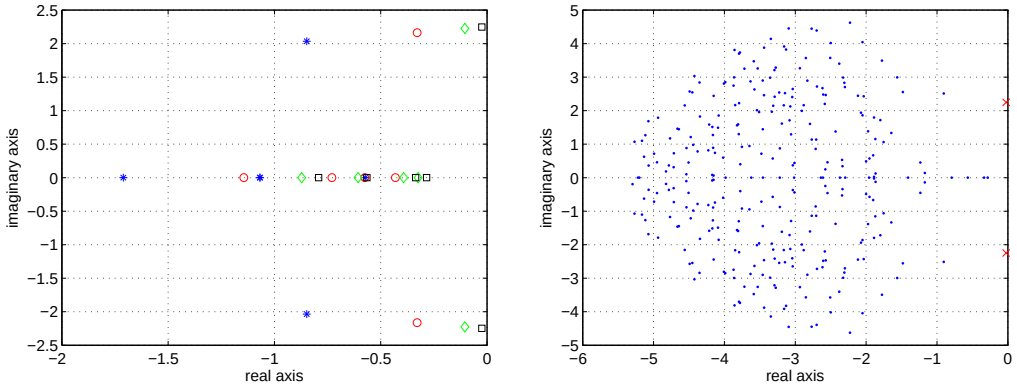
(a) $Re = 2000$ (b) $Re = 4000$

(c) $Re = 6000$ (d) $Re = 7800$

Figure 4.3: Comparison of the standard Krylov method and RKSM in solving (4.8) for driven-cavity flow

300 eigenvalues of (2.2) with smallest modulus at $Re = 350$. As for the previous example, as the Reynolds number increases, the six rightmost eigenvalues all move towards the imaginary axis, and the rightmost eigenvalue changes from being real (at $Re = 100$) to complex (at $Re = 200, 300, 350$). The rightmost pair of eigenvalues of (2.2) cross the imaginary axis and the steady-state solution to (2.1) loses its stability at $Re \approx 373$ (see Table A.3).

We again apply Algorithm 8 to estimate the distance between the rightmost eigenvalue of (2.2) and the imaginary axis for the four Reynolds numbers mentioned



(a) (b)

Figure 4.4: (a) The six rightmost eigenvalues for flow over an obstacle at different Reynolds numbers ($*$: $Re = 100$, $\circ$: $Re = 200$, $\diamondsuit$: $Re = 300$, $\square$: $Re = 350$). (b) The 300 eigenvalues with smallest modulus at $Re = 350$ ($\times$: the rightmost eigenvalues).

above. The results are reported in Table 4.4 (see Table 4.2 for notation). The stopping criteria for both Algorithm 8 and the Lyapunov solve (2.24) remain unchanged, *i.e.*, (4.6) and (4.7). For all four Reynolds numbers, Algorithm 8 converges rapidly. In fact, we will show in section 4.2.4 that if the Lyapunov equation (4.8) is solved more accurately, Algorithm 8 will converge in two iterations in all four cases as observed in the previous example. Again we compare the performance of the standard Krylov method and RKSM in solving (4.8). As for the cavity flow, the standard Krylov method needs a significantly larger subspace than RKSM to compute a solution of the same accuracy (see Figure 4.5).

4.2.3 Example 3: Double-Diffusive Convection Problem

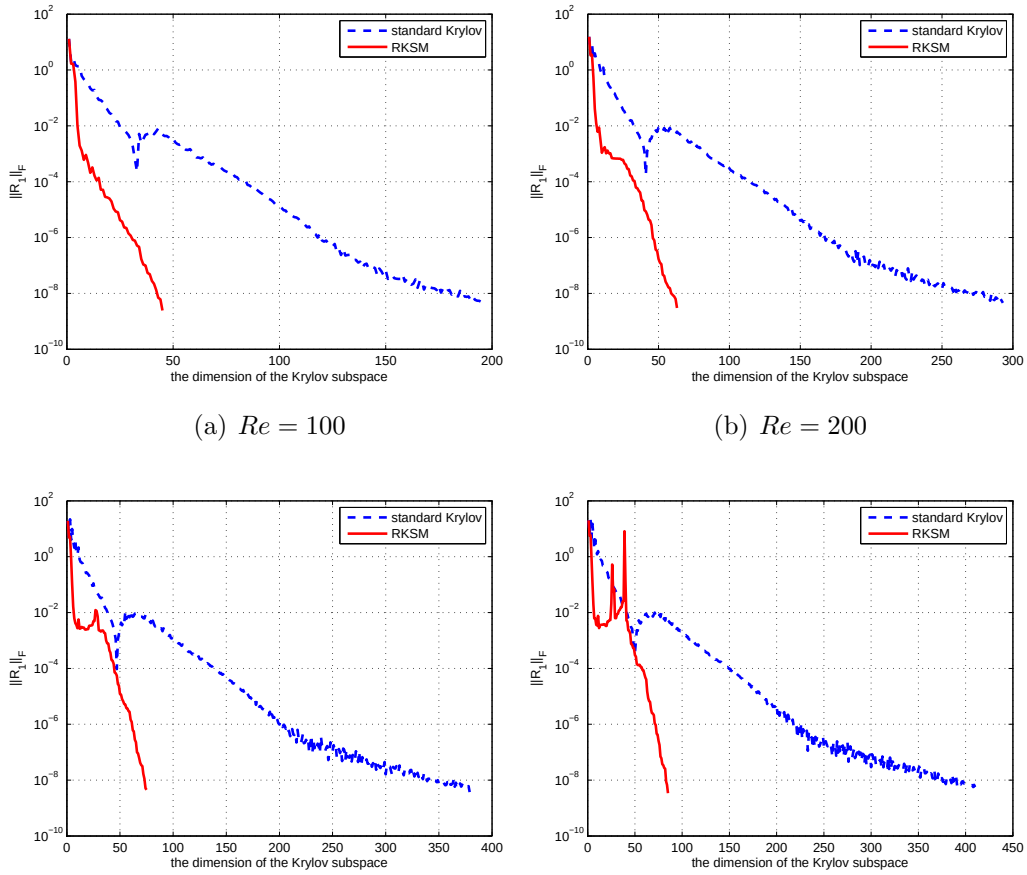
This is a model of the effects of convection and diffusion on two solutions in a box heated at one boundary (see Chapter 8 of [49]). The governing equations use

Table 4.4: Algorithm 8 applied to flow over an obstacle

ℓ	$\lambda^{(\ell)}$	$\mu^{(\ell)}$	$\|R_\ell^{eig}\|_F$	$\|R_\ell^{lyap}\|_F$	d_ℓ
		Re=100			
1	-2.42460	2.42460	1.15123e+1	2.44638e-09	45
2	0.57285	-0.57285	1.28322e-4	1.15950e-11	22
3	0.57285	-0.57285	4.86146e-6	1.64039e-09	18
4	0.57285	-0.57285	9.49881e-7	5.11488e-09	10
5	0.57285	-0.57285	1.35238e-7	5.65183e-09	4
6	0.57285	-0.57285	2.30716e-8	8.18398e-10	4
7	0.57285	-0.57285	8.43416e-9	—	—
		Re=200			
1	-2.45074	2.45074	1.16834e+1	3.00582e-09	63
2	0.32884	-0.32884+2.16396i	3.86737e-5	2.20976e-10	86
3	0.32884	-0.32884+2.16393i	1.30869e-8	2.04006e-10	46
4	0.32884	-0.32884+2.16393i	1.47390e-9	—	—
		Re=300			
1	-2.47804	2.47804	1.18371e+01	4.49864e-09	75
2	0.10405	-0.10405+2.22643i	7.59831e-07	3.60446e-10	86
3	0.10405	-0.10405+2.22643i	2.18881e-10	—	—
		Re=350			
1	-2.49317	2.49317	1.19385e+01	3.40780e-09	85
2	0.02411	-0.02411+2.24736i	2.80626e-08	3.84715e-10	90
3	0.02411	-0.02411+2.24736i	1.46747e-11	—	—

Boussinesq approximation and are given in [7] and [9]. Linear stability analysis of this problem is considered in [17]. The imaginary parts of the rightmost eigenvalues of (2.2) near the critical point $(\overline{u}_c, \alpha_c)$ have fairly large magnitude, and as a result, the rightmost eigenvalues are further away from zero than many of the real eigenvalues close to the imaginary axis. Conventional methods, such as IRA with a zero shift, tend to converge to the real eigenvalues close to the imaginary axis instead of the rightmost pair.

We consider an artificial version $\mathbf{A}x = \mu x$ of this problem, where $\mathbf{A}$ is tridiagonal of order $n = 10,000$ with eigenvalues $\mu_{1,2} = -0.05 \pm 25i$ and $\mu_j = -(j-1)\cdot 0.1$ for all $3 \leq j \leq n$. The 300 eigenvalues of $\mathbf{A}$ with smallest modulus are plotted in



(a) $Re = 100$ (b) $Re = 200$

(c) $Re = 300$ (d) $Re = 350$

Figure 4.5: Comparison of the standard Krylov method and RKSM for solving (4.8) in flow over an obstacle

Figure 4.6 (left). A similar problem is studied in [28]. If we use the Matlab function 'eigs' with zero shift to compute its rightmost eigenvalues, at least 251 eigenvalues of $\mathbf{A}$ have to be computed to ensure that $\mu_{1,2}$ will be found. This approach requires a minimum 502 linear solves under the default setting of 'eigs', and again in practice many more will be needed. We apply Algorithm 8 to this problem (with the same stopping criteria for the inner and outer iterations as in the previous two examples) and the results are reported in Table 4.5. It converges in just 3 iterations, requiring 90 linear solves to solve the two Lyapunov equations to desired accuracy. As

in the previous examples, RKSM needs a Krylov subspace of significantly smaller dimension than the standard Krylov method (see Figure 4.6 (right)).

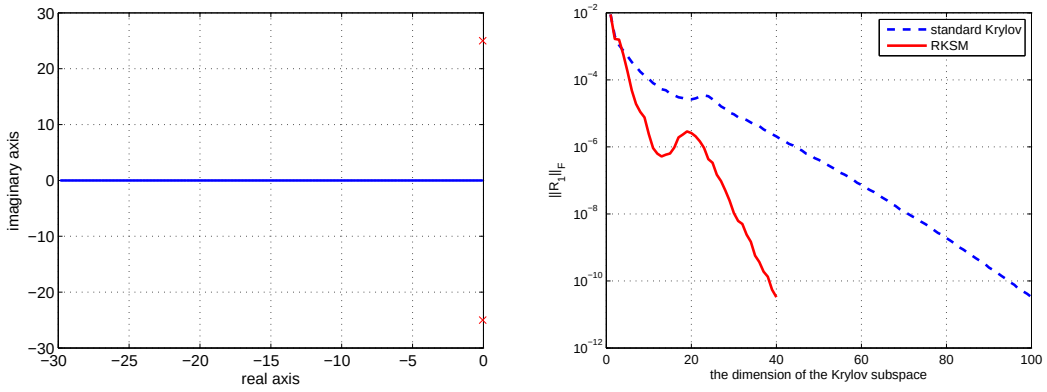


Figure 4.6: Left: the 300 eigenvalues with smallest modulus ($\times$: the rightmost eigenvalues). Right: Comparison of the standard Krylov method and RKSM for solving (4.8) in double-diffusive convection problem

Table 4.5: Algorithm 8 applied to double-diffusive convection problem

ℓ	$\lambda^{(\ell)}$	$\mu^{(\ell)}$	$\|R_\ell^{eig}\|_F$	$\|R_\ell^{lyap}\|_F$	d_ℓ
1	109.973	-109.973	4.33472e+00	3.28342e-11	40
2	0.05000	-0.05000+25.0000i	4.76830e-08	3.01965e-12	50
3	0.05000	-0.05000+25.0000i	1.56010e-13	—	—

4.2.4 Analysis of the Convergence of Algorithm 8

In the numerical experiments, we have shown that Algorithm 8 converges rapidly. In particular, it converges in just two iterations in many cases, for example, the driven-cavity flow at $Re = 2000, 4000, 6000, 7800$. In other words, only one Lyapunov solve (4.8) is needed to obtain an eigenvalue estimate of desired accuracy in these cases. The analysis below provides some insight into this fast convergence.

We introduce some notation to be used in the analysis. Assuming (2.2) and therefore (4.1) have complete sets of eigenvectors, let $\{(\lambda_k, z_k)\}_{k=1}^{n^2}$ denote the eigen-

pairs of (4.1), where $\|z_k\|_2 = 1$ and $\{\lambda_k\}$ have increasing moduli, *i.e.*, $|\lambda_{k_1}| \leq |\lambda_{k_2}|$ if $k_1 < k_2$. By Theorem 4.1, for each $1 \leq k \leq n^2$, there exist $1 \leq i, j \leq n$ such that $\lambda_k = \lambda_{i,j} = -\frac{1}{2}(\mu_i + \mu_j)$ and $z_k = z_{i,j} = x_i \otimes x_j$.[2] Let $Z_k = x_j x_i^T$, *i.e.*, $vec(Z_k) = z_k$, so that $\{(\lambda_k, Z_k)\}_{k=1}^{n^2}$ are the eigenpairs of (4.2). In addition, let $L_p = \{\lambda_1, \lambda_2, \ldots, \lambda_p\}$ contain the p eigenvalues of (4.1) with smallest modulus, and let $E_p = \{\mu_{i_1}, \mu_{i_2}, \ldots, \mu_{i_d}\}$ be the smallest subset of eigenvalues of $\mathbf{A}x = \mu\mathbf{M}x$ that satisfies the following: for any $\lambda_k \in L_p$, there exist $\mu_{i_s}, \mu_{i_t} \in E_p$ such that $\lambda_k = \lambda_{i_s,i_t}$. Let $X_p = \{x_{i_1}, x_{i_2}, \ldots, x_{i_d}\}$ hold the eigenvectors of $\mathbf{A}x = \mu\mathbf{M}x$ associated with E_p. For a concrete example, consider again the 4×4 example in section 2. From Table 4.1, $L_7 = \{\lambda_{1,2}, \lambda_{2,1}, \lambda_{3,3}, \lambda_{1,3}, \lambda_{3,1}, \lambda_{2,3}, \lambda_{3,2}\}$, $E_7 = \{\mu_1, \mu_2, \mu_3\}$, and $X_7 = \{x_1, x_2, x_3\}$.

We first look at standard inverse iteration applied to (4.1). Let the starting guess be $z^{(1)} = v \otimes v \in \mathbb{R}^{n^2}$, where $v \in \mathbb{R}^n$ is a random vector of unit norm. Since $\{z_k\}_{k=1}^{n^2}$ are linearly independent, $z^{(1)}$ can be written as $\sum_{k=1}^{n^2} \xi_k z_k$ with $\xi_k \in \mathbb{C}$ (assume $\xi_k \neq 0$ for any k). Note that the coefficients $\{\xi_k\}_{k=1}^{n^2}$ have the following properties: if $z_{k_1} = \overline{z_{k_2}}$, then $\xi_{k_1} = \overline{\xi_{k_2}}$; moreover, if $z_{k_1} = x_i \otimes x_j$ and $z_{k_2} = x_j \otimes x_i$ for some pair (i, j), then $\xi_{k_1} = \xi_{k_2}$. The first property is due to the fact that $z^{(1)}$ is real, and the second one is a result of the special tensor structure of $z^{(1)}$. In the first step of inverse iteration, we solve the linear system

$$\Psi_1 y_1 = (-\Psi_0) z^{(1)}. \tag{4.9}$$

[2]Both sets of symbols $\{\lambda_{i,j}, z_{i,j}\}_{i,j=1}^{n}$ and $\{\lambda_k, z_k\}_{k=1}^{n^2}$ denote the eigenpairs of (4.1). The double subscripts indicate the special structure of the eigenpairs, whereas the single subscripts arrange the eigenvalues in ascending order of their moduli. Our choice between the two notations depends on the context.

The solution to (4.9) is $y_1 = \Psi_1^{-1}(-\Psi_0)z^{(1)} = \sum_{k=1}^{n^2} \frac{\xi_k}{\lambda_k} z_k$. Let y_1^p be a truncated approximation of y_1 consisting of its p dominant components, *i.e.*, $y_1^p = \sum_{k=1}^{p} \frac{\xi_k}{\lambda_k} z_k$ for some $p \ll n^2$.

Next, we consider Algorithm 8 applied to (4.2). Let the starting guess be $Z^{(1)} = vv^T$, where v is the vector that determines the starting vector for standard inverse iteration. At step 2.4 of the first iteration, we solve the Lyapunov equation (4.8) where $vec\left(P_1 C_1 P_1^T\right) = vec\left(-2SZ^{(1)}S^T\right) = (-\Psi_0)z^{(1)}$ (see (4.4)) and $vec(Y_1) = y_1$, *i.e.*,

$$Y_1 = \sum_{k=1}^{n^2} \frac{\xi_k}{\lambda_k} Z_k. \tag{4.10}$$

Let Y_1^p denote the truncation of Y_1 that satisfies $vec\left(Y_1^p\right) = y_1^p$, *i.e.*, $Y_1^p = \sum_{k=1}^{p} \frac{\xi_k}{\lambda_k} Z_k$. Assume p is chosen such that if $\lambda_{i,j} \in L_p$, then $\overline{\lambda_{i,j}}, \lambda_{j,i} \in L_p$ as well. Under this assumption and by properties of the coefficients $\{\xi_k\}_{k=1}^{n^2}$, Y_1^p is real and symmetric and can be written as $Y_1^p = \mathcal{U}\mathcal{G}\mathcal{U}^T$ where $\mathcal{U} \in \mathbb{R}^{n\times d}$ is an orthonormal matrix whose columns span X_p and $\mathcal{G} \in \mathbb{R}^{d\times d}$ is symmetric. Recall from section 2 that the target eigenpair of (4.2) sought by Algorithm 8 is $\left(\lambda_1, \mathcal{V}\mathcal{D}\mathcal{V}^T\right)$, where $\lambda_1 = -Re(\mu_1)$, and $\mathcal{V} \in \mathbb{R}^{n\times q}$ (with $q = 1$ or 2) is x_1 (if μ_1 is real) or an orthonormal matrix whose columns span $\{x_1, \overline{x_1}\}$ (if μ_1 is not real). Since $(\lambda_1, \mathcal{V}\mathcal{D}\mathcal{V}^T)$ is an eigenpair of (4.2),

$$S\left(\mathcal{V}\mathcal{D}\mathcal{V}^T\right) + \left(\mathcal{V}\mathcal{D}\mathcal{V}^T\right)S^T + \lambda_1\left(2S\left(\mathcal{V}\mathcal{D}\mathcal{V}^T\right)S^T\right) = 0. \tag{4.11}$$

For any $p \geq 1$, $x_1, \overline{x_1} \in X_p$. Thus, $\mathcal{U}$ can be taken to have the form $\mathcal{U} = [\mathcal{V}, \mathcal{V}^\perp]$ where $\mathcal{V}^T\mathcal{V}^\perp = \mathbf{0}$.

Assume that step 2.4 of the first iteration of Algorithm 9 produces an approximate solution to (4.8) of the form Y_1^p by some means (that is, the approximate solution consists of the p dominant terms of (4.10) where p satisfies the assumption above). Then at step 2.1 (rank reduction) of the second iteration, we solve the $d \times d$ projected eigenvalue problem

$$\left(\mathcal{U}^T S \mathcal{U}\right) \widetilde{Z} + \widetilde{Z} \left(\mathcal{U}^T S \mathcal{U}\right)^T + \widetilde{\lambda} \left(2 \left(\mathcal{U}^T S \mathcal{U}\right) \widetilde{Z} \left(\mathcal{U}^T S \mathcal{U}\right)^T\right) = 0 \tag{4.12}$$

for the eigenvalue with smallest modulus, $\widetilde{\lambda}_1$, and its associated real, symmetric and rank-q eigenvector $\widetilde{Z}_1$ ($q = 1$ or 2). We then obtain an estimate of the target eigenpair of (4.2), namely $\left(\widetilde{\lambda}_1, \mathcal{U}\widetilde{Z}_1\mathcal{U}^T\right)$. It can be shown that this estimate is in fact exact, that is, $\left(\widetilde{\lambda}_1, \mathcal{U}\widetilde{Z}_1\mathcal{U}^T\right) = (\lambda_1, \mathcal{V}\mathcal{D}\mathcal{V}^T)$.

Theorem 4.3. *If*

$$D_1 = \begin{bmatrix} \mathcal{D} & \\ & \mathbf{0} \end{bmatrix}_{d \times d}, \tag{4.13}$$

then (λ_1, D_1) *is an eigenpair of* (4.12).

Proof. Left-multiply (4.11) by $\mathcal{U}^T$ and right-multiply by $\mathcal{U}$:

$$\mathcal{U}^T S(\mathcal{V}\mathcal{D}\mathcal{V}^T)\mathcal{U} + \mathcal{U}^T(\mathcal{V}\mathcal{D}\mathcal{V}^T)S^T\mathcal{U} + \lambda_1 2\mathcal{U}^T S(\mathcal{V}\mathcal{D}\mathcal{V}^T)S^T\mathcal{U} = 0.$$

Since $\mathcal{U} = [\mathcal{V}, \mathcal{V}^\perp]$, it follows that $\mathcal{V}\mathcal{D}\mathcal{V}^T = \mathcal{U}D_1\mathcal{U}^T$. Therefore,

$$\left(\mathcal{U}^T S \mathcal{U}\right) D_1 + D_1 \left(\mathcal{U}^T S^T \mathcal{U}\right) + \lambda_1 \left(2 \left(\mathcal{U}^T S \mathcal{U}\right) D_1 \left(\mathcal{U}^T S^T \mathcal{U}\right)\right) = 0.$$

□

Proposition 4.4. *The eigenvalue of* (4.12) *with smallest modulus is* $\widetilde{\lambda}_1 = \lambda_1$.

Proof. Let $\widetilde{S} = \mathcal{U}^T S \mathcal{U}$. Since the columns of $\mathcal{U}$ span X_p, the eigenvalues of $\widetilde{S}$ are $\mu_{i_1}^{-1}$, $\mu_{i_2}^{-1}$, …, $\mu_{i_d}^{-1}$. Let $\widetilde{\Psi}_1 = \widetilde{S} \otimes I + I \otimes \widetilde{S}$ and $\widetilde{\Psi}_0 = 2\widetilde{S} \otimes \widetilde{S}$. By Theorem 4.1, the eigenvalues of the $d^2 \times d^2$ eigenvalue problem

$$\left(\widetilde{\Psi}_1 + \widetilde{\lambda}\widetilde{\Psi}_0\right)\widetilde{z} = 0 \tag{4.14}$$

are $\widetilde{\lambda}_{s,t} = -\frac{1}{2}(\mu_{i_s} + \mu_{i_t})$ for any $1 \leq s, t \leq d$. Since $\mu_1, \overline{\mu_1} \in E_p$, Theorem 4.2 implies that the eigenvalue of (4.14) with smallest modulus is $\widetilde{\lambda}_1 = \lambda_1$, the eigenvalue of (4.1) with smallest modulus. Since (4.12) and (4.14) have the same eigenvalues, the eigenvalue of (4.12) with smallest modulus is λ_1 as well. □

Recall that we solve the Lyapunov equation (4.8) using an iterative solver such as RKSM, which produces a real, symmetric approximate solution Y_1^{approx}. By Theorem 4.3 and Proposition 4.4, if $Y_1^{approx} = Y_1^p$, then after the rank-reduction step in the second iteration of Algorithm 8, we obtain the exact eigenpair $\left(\lambda_1, \mathcal{V}\mathcal{D}\mathcal{V}^T\right)$ of (4.2) that we are looking for. In reality, it is unlikely that the approximate solution Y_1^{approx} we compute will be exactly Y_1^p. However, since Y_1^p consists of the p dominant terms of the exact solution (4.10), if Y_1^{approx} is accurate enough, then $Y_1^{approx} \approx Y_1^p$ for some p.

This analysis suggests that the eigenvalue residual norm $\|R_2^{eig}\|_F$ can be made arbitrarily small as long as the residual norm of the Lyapunov system $\|R_1^{lyap}\|_F$ is small enough. Therefore, we propose the following modified version of Algorithm 8

(given $\tau_{lyap}, \tau_{eig} > 0$):

Algorithm 9: Modified Lyapunov inverse iteration for (4.2)

1. Given $\mathbf{V}_1 \in \mathbb{R}^n$ with $\|\mathbf{V}_1\|_2 = 1$. Set $\ell = 1$ and $firsttry = \texttt{true}$.
2. Rank reduction: compute $\widetilde{S} = \mathbf{V}_\ell^T S \mathbf{V}_\ell$ and solve for the eigenvalue $\widetilde{\lambda}_1$ of (4.3) with smallest modulus and its eigenvector $\widetilde{Z}_1 = \widetilde{\mathcal{V}}\widetilde{\mathcal{D}}\widetilde{\mathcal{V}}^T$.
3. Set $\left(\lambda^{(\ell)}, Z^{(\ell)}\right) = \left(\widetilde{\lambda}_1, \mathcal{V}_\ell \widetilde{D} \mathcal{V}_\ell^T\right)$ where $\mathcal{V}_\ell = \mathbf{V}_\ell \widetilde{\mathcal{V}}$, and compute $\|R_\ell^{eig}\|_F$.
4. While $\|R_\ell^{eig}\|_F > \tau_{eig}$:
   - 4.1 if $firsttry$
     compute an approximate solution $Y_1^{approx} = \mathbf{V}_2 D_2 \mathbf{V}_2^T$ to (4.8) such that $\|R_1^{lyap}\|_F < \tau_{lyap} \cdot \|C_1\|_F$; set $\ell = 2$ and $firsttry = \texttt{false}$;
   - 4.2 else
     solve (4.8) more accurately and update $\mathbf{V}_2$;
   - 4.3 repeat steps 2 and 3 to compute $\lambda^{(\ell)}$, $Z^{(\ell)}$ and $\|R_\ell^{eig}\|_F$.

In this algorithm, if $\left(\lambda^{(2)}, Z^{(2)}\right)$ is not an accurate enough eigenpair ($\|R_2^{eig}\|_F \geq \tau_{eig}$), this is fixed by improving the accuracy of the Lyapunov system (4.8). The discussion above shows that this will be enough to produce an accurate eigenpair. Moreover, it is possible to get an improved solution to (4.8) by augmenting the solution we have in hand. Assume that at step 4.1 of Algorithm 9, we compute an approximate solution $Y_1^{approx} = \mathbf{V}_2 D_2 \mathbf{V}_2^T$ to (4.8) where the columns of $\mathbf{V}_2$ span the Krylov subspace $\mathcal{K}_m(S, P_1)$ (see (3.7) for definition), and then obtain an iterate $\left(\lambda^{(2)}, Z^{(2)}\right)$ in steps 2 and 3. If $\|R_2^{eig}\|_F \geq \tau_{eig}$, we perform one more block Arnoldi step to extend the existing Krylov subspace to $\mathcal{K}_{m+1}(S, P_1)$, obtain a new approximate solution $Y_1^{approx} = \mathbf{V}_2 D_2 \mathbf{V}_2^T$ to (4.8) where the columns of V_1 now span the augmented Krylov subspace $\mathcal{K}_{m+1}(S, P_1)$, and check convergence in steps 2 and 3 again. We keep extending the Krylov subspace at our disposal until the outer iteration converges to the desired tolerance τ_{eig}.

We test Algorithm 9 on Example 2 and Example 3, where Algorithm 8 con-

verged in more than two iterations (see Tables 4.4 and 4.5). As in the previous experiments, we choose $\tau_{lyap} = 10^{-9}$ and $\tau_{eig} = 10^{-8}$. The results are reported in Tables 4.6 and 4.7, from which it can be seen that if (4.8) is solved accurately enough, Lyapunov inverse iteration converges to the desired tolerance in only two iterations, as observed for the driven-cavity flow (see Table 4.3). In other words, it requires only one Lyapunov solve (4.8). By comparing Tables 4.5 and 4.7, for example, we can see that in order to compute an accurate enough approximate solution Y_1^{approx} to (4.8), the dimension of the Krylov subspace used must be increased from 40 to 43.

Table 4.6: Algorithm 9 applied to flow over an obstacle

ℓ	$\lambda^{(\ell)}$	$\mu^{(\ell)}$	$\|R_\ell^{eig}\|_F$	$\|R_\ell^{lyap}\|_F$	d_ℓ
			Re=100		
1	-2.42460	2.42460	1.15123e+1	**4.45806e-12**	**65**
2	0.57285	-0.57285	7.98509e-9	—	—
			Re=200		
1	-2.45074	2.45074	1.16834e+1	**2.79438e-12**	**83**
2	0.32884	-0.32884+2.16393i	7.67144e-9	—	—
			Re=300		
1	-2.47804	2.47804	1.18371e+1	**1.35045e-10**	**86**
2	0.10405	-0.10405+2.22643i	6.10287e-9	—	—
			Re=350		
1	-2.49317	2.49317	1.19385e+1	**1.07068e-09**	**88**
2	0.02411	-0.02411+2.24736i	7.16343e-9	—	—

Table 4.7: Algorithm 9 applied to double-diffusive convection problem

ℓ	$\lambda^{(\ell)}$	$\mu^{(\ell)}$	$\|R_\ell^{eig}\|_F$	$\|R_\ell^{lyap}\|_F$	d_ℓ
1	109.973	-109.973	4.33472e+0	**4.71359e-12**	**43**
2	0.05000	-0.05000+25.0000i	6.88230e-9	—	—

4.3 Computing a Few Rightmost Eigenvalues

In section 4.1, we showed that when all the eigenvalues of (2.2) lie in the left half of the complex plane, the distance between the rightmost eigenvalue and the imaginary axis, $-Re(\mu_1)$, is the eigenvalue of (4.2) with smallest modulus. As a result, this eigenvalue can be computed by Lyapunov inverse iteration, which also gives us estimates of the rightmost eigenvalue of (2.2). In section 4.2, various numerical experiments demonstrate the robustness and efficiency of the Lyapunov inverse iteration applied to (4.2). In particular, we showed in section 4.2.4 that if the first Lyapunov equation (4.8) is solved accurately enough, then Lyapunov inverse iteration will converge in only two steps. As seen in sections 4.2.1 and 4.2.2, when we march along the solution path $\mathcal{S}$, it may be the case that an eigenvalue that is not the rightmost moves towards the imaginary axis rapidly, becomes the rightmost eigenvalue at some point and eventually crosses the imaginary axis first, causing instability in the steady-state solution. Therefore, besides the rightmost eigenvalue, it is helpful to monitor a few other eigenvalues in the rightmost part of the spectrum as well. In this section, we show how Lyapunov inverse iteration can be applied repeatedly in combination with deflation to compute k rightmost eigenvalues of (2.2), where $1 < k \ll n$.

We continue to assume that we are at a point $(\overline{u}_0, \alpha_0)$ in the stable regime of the solution path $\mathcal{S}$ and that the eigenvalue problem $\mathbf{A}x = \mu\mathbf{M}x$ with $\mathbf{A} = \mathcal{J}(\alpha_0)$ has a complete set of eigenvectors $\{x_i\}_{i=1}^n$. For any $i \leq k$, we also assume the following (as in assumptions **(a1)** and **(a2)** in section 2):

(**a1′**) if $Re(\mu_j) = Re(\mu_i)$ and $j \neq i$, then $\mu_j = \overline{\mu_i}$; and

(**a2′**) μ_i is a simple eigenvalue.

Let $\mathcal{E}_t = \{\mu_1, \mu_2, \ldots, \mu_t\}$ be the set containing t rightmost eigenvalues of $\mathbf{A}x = \mu\mathbf{M}x$ and $\mathcal{X}_t = [x_1, x_2, \ldots, x_t] \in \mathbb{C}^{n\times t}$ be the matrix that holds the t corresponding eigenvectors. Here t is chosen such that $t < k$ and if $\mu_i \in \mathcal{E}_t$, then $\overline{\mu_i} \in \mathcal{E}_t$ as well. We will show that given $\mathcal{X}_t$, we can use the methodology described in section 2 to find $-Re(\mu_{t+1})$, that is, that $-Re(\mu_{t+1})$ is the eigenvalue with smallest modulus of a certain $n^2 \times n^2$ eigenvalue problem with a Kronecker structure like that of (4.1), and it can be computed using Lyapunov inverse iteration.

Lemma 4.5. *Assume all the eigenvalues of* $\mathbf{A}x = \mu\mathbf{M}x$ *lie in the left half of the complex plane. Then in the subset* $\{\lambda_{i,j}\}_{i,j>t}$ *of all the eigenvalues of* (4.1)*, the one with smallest modulus is* $-Re(\mu_{t+1})$*.*

Proof. If μ_{t+1} is real, then $-Re(\mu_{t+1}) = \lambda_{t+1,t+1}$. If μ_{t+1} is not real, by assumptions **(a1′)**, **(a2′)** and the choice of t, $\mu_{t+2} = \overline{\mu_{t+1}}$, which implies that $-Re(\mu_{t+1}) = \lambda_{t+1,t+2} = \lambda_{t+2,t+1}$. The rest of the proof is very similar to that of Theorem 4.2. □

Consequently, if we can formulate a problem with a Kronecker structure like that of (4.1) whose eigenvalues are $\{\lambda_{i,j}\}_{i,j>t}$, then $-Re(\mu_{t+1})$ can be computed by Lyapunov inverse iteration applied to this problem. We will show how such a problem can be concocted and establish some of its properties that are similar to those of (4.1).

Let Θ_t be the diagonal matrix whose diagonal elements are $\mu_1^{-1}, \mu_2^{-1}, \ldots, \mu_t^{-1}$, so that $S\mathcal{X}_t = \mathcal{X}_t\Theta_t$. Since $\mathbf{A}x = \mu\mathbf{M}x$ has a complete set of eigenvectors, there exists an orthonormal matrix $Q_t \in \mathbb{R}^{n\times t}$ such that $\mathcal{X}_t = Q_tG_t$, where $G_t \in \mathbb{C}^{t\times t}$ is nonsingular. Let

$$\widehat{S} = \left(I - Q_tQ_t^T\right)S, \quad \widehat{\Psi}_1 = \widehat{S}\otimes I + I\otimes\widehat{S}, \quad \text{and} \quad \widehat{\Psi}_0 = 2\widehat{S}\otimes\widehat{S}.$$

We claim that the distance between μ_{t+1} and the imaginary axis, $-Re(\mu_{t+1})$, is the eigenvalue of

$$\left(\widehat{\Psi}_1 + \lambda\widehat{\Psi}_0\right)z = 0, \quad z \in Range\left(\widehat{\Psi}_0\right) \tag{4.15}$$

with smallest modulus. To prove this claim, we first study the eigenpairs of $\widehat{S}$.

Lemma 4.6. *The matrix $I - Q_tQ_t^T$ where Q_t is defined above and $I \in \mathbb{R}^{n\times n}$ is the identity matrix has the following properties:*

1. $\left(I - Q_tQ_t^T\right)Q_t = \mathbf{0}$;
2. $\left(I - Q_tQ_t^T\right)^i = \left(I - Q_tQ_t^T\right)$ *for any integer* $i \geq 1$;
3. $\left(I - Q_tQ_t^T\right)^i S\left(I - Q_tQ_t^T\right)^j = \left(I - Q_tQ_t^T\right)S$ *for any integers* $i, j \geq 1$.

Proof. The first two properties hold for any orthonormal matrix and the proof is omitted here. To prove the third property, we first show that

$$\left(I - Q_tQ_t^T\right)S\left(I - Q_tQ_t^T\right) = \left(I - Q_tQ_t^T\right)S.$$

Since $S\mathcal{X}_t = \mathcal{X}_t\Theta_t$ and $\mathcal{X}_t = Q_tG_t$, $SQ_tQ_t^T = Q_tG_t\Theta_tG_t^{-1}Q_t^T$ (G_t is invertible). Thus,

$$\begin{aligned}\left(I - Q_tQ_t^T\right) S \left(I - Q_tQ_t^T\right) &= \left(I - Q_tQ_t^T\right) S - \left(I - Q_tQ_t^T\right) SQ_tQ_t^T \\ &= \left(I - Q_tQ_t^T\right) S - \left(I - Q_tQ_t^T\right) Q_tG_t\Theta_tG_t^{-1}Q_t^T \\ &= \left(I - Q_tQ_t^T\right) S\end{aligned}$$

by the first property. This together with the second property establishes the third property. □

Lemma 4.7. *Let $\widehat{\theta}_i = 0$ for $i \leq t$ and $\widehat{\theta}_i = \frac{1}{\mu_i}$ for $i > t$. Let $\widehat{x}_i = x_i$ for $i \leq t$ and $\widehat{x}_i = \left(I - Q_tQ_t^T\right) x_i$ for $i > t$. Then $\left(\widehat{\theta}_i, \widehat{x}_i\right)$ $(i = 1, 2, \ldots, n)$ are the eigenpairs of $\widehat{S}$.*

Proof. Let g_i be the i^{th} column of G_t. If $i \leq t$, $x_i = Q_tg_i$, thus

$$\widehat{S}x_i = \left(I - Q_tQ_t^T\right) SQ_tg_i = \left(I - Q_tQ_t^T\right) Q_tG_t\Theta_tG_t^{-1}g_i = \mathbf{0}$$

by the first property in Lemma 4.6. If $i > t$,

$$\begin{aligned}\widehat{S}\left(I - Q_tQ_t^T\right) x_i = \left(I - Q_tQ_t^T\right) S \left(I - Q_tQ_t^T\right) x_i &= \left(I - Q_tQ_t^T\right) Sx_i \\ &= \frac{1}{\mu_i}\left(I - Q_tQ_t^T\right) x_i\end{aligned}$$

by the third property in Lemma 4.6. □

Knowing the eigenpairs of $\widehat{S}$, we can find the eigenpairs of $\widehat{\Psi}_0$ and $\widehat{\Psi}_1$ with no difficulty.

Lemma 4.8. *The eigenvalues of $\widehat{\Psi}_1$ are*

1. 0, *if* $i, j \leq t$;

2. $\frac{1}{\mu_i}$, *if* $i > t$ *and* $j \leq t$;

3. $\frac{1}{\mu_j}$, *if* $i \leq t$ *and* $j > t$;

4. $\frac{1}{\mu_i} + \frac{1}{\mu_j}$, *if* $i, j > t$.

The eigenvalues of $\widehat{\Psi}_0$ *are*

1. 0, *if* $i \leq t$ *or* $j \leq t$;

2. $\frac{2}{\mu_i \mu_j}$, *if* $i, j > t$.

Moreover, for each eigenvalue of $\widehat{\Psi}_0$ *or* $\widehat{\Psi}_1$, *there are eigenvectors associated with it given by* $\widehat{z}_{i,j} = \widehat{x}_i \otimes \widehat{x}_j$ *and* $\widehat{z}_{j,i} = \widehat{x}_j \otimes \widehat{x}_i$.

Proof. See the proof of Theorem 4.1. □

Under the assumption that $\mathbf{A}x = \mu \mathbf{M}x$ has a complete set of eigenvectors, $\widehat{\Psi}_0$ also has a complete set of eigenvectors $\{\widehat{z}_{i,j}\}_{i,j=1}^{n}$. By Lemma 4.8, *Range* $\left(\widehat{\Psi}_0\right) =$ *span* $\{\widehat{z}_{i,j}\}_{i,j>t}$.

Theorem 4.9. *The eigenvalues of* (4.15) *are* $\{\lambda_{i,j}\}_{i,j>t}$. *For any* $\lambda_{i,j}$ *with* $i, j > t$, *there are eigenvectors* $\widehat{z}_{i,j}$ *and* $\widehat{z}_{j,i}$ *associated with it.*

Proof. The proof follows immediately from Lemma 4.8 and the proof of Theorem 4.1. □

Theorem 4.10. *Assume all the eigenvalues of* $\mathbf{A}x = \mu \mathbf{M}x$ *lie in the left half of the complex plane. Then the eigenvalue of* (4.15) *with smallest modulus is* $-Re(\mu_{t+1})$.

Proof. By Theorem 4.9, it suffices to show that $|Re(\mu_{t+1})| \leq |\lambda_{i,j}|$ for any $i, j > t$, which is true by Lemma 4.5. □

If we can restrict the search space of eigenvectors to $Range\left(\widehat{\Psi}_0\right)$, we can apply inverse iteration to $\widehat{\Psi}_1 z = \lambda\left(-\widehat{\Psi}_0\right) z$ to compute $-Re(\mu_{t+1})$. Let

$$\mathbb{P}_t = \{Z \in \mathbb{C}^{n\times n} | Z = \left(I - Q_t Q_t^T\right) X \left(I - Q_t Q_t^T\right) \ where\ X \in \mathbb{C}^{n\times n}\}.$$

Since

$$Range\left(\widehat{\Psi}_0\right) = span\left\{\widehat{x}_i \otimes \widehat{x}_j\right\}_{i,j>t} = span\left\{\left(I - Q_t Q_t^T\right) x_i \otimes \left(I - Q_t Q_t^T\right) x_j\right\}_{i,j>t},$$

if $Z \in \mathbb{P}_t$, then $z = vec(Z) \in Range\left(\widehat{\Psi}_0\right)$, and *vice versa*. Therefore, (4.15) can be rewritten in the form of a matrix equation,

$$\widehat{S}Z + Z\widehat{S}^T + \lambda\left(2\widehat{S}Z\widehat{S}^T\right) = 0,\ Z \in \mathbb{P}_t. \tag{4.16}$$

By Theorem 4.10, $-Re(\mu_{t+1})$ is the eigenvalue of (4.16) with smallest modulus. As in section 2, under certain conditions, we can show that $-Re(\mu_{t+1})$ is an eigenvalue of (4.16) with a unique, real, symmetric and low-rank eigenvector. Let $\mathbb{P}_t^s = \left\{Z \in \mathbb{P}_t | Z = Z^T\right\}$ be the subspace of $\mathbb{P}_t$ consisting of symmetric matrices. As a result of assumptions **(a1′)** and **(a2′)**, when the eigenspace of (4.16) is restricted to $\mathbb{P}_t^s$, $-Re(\mu_{t+1})$ is an eigenvalue of (4.16) that has the unique (up to a scalar multiplier),

real and symmetric eigenvector

$$\left(I - Q_t Q_t^T\right) x_{t+1} x_{t+1}^T \left(I - Q_t Q_t^T\right) \text{ or } \left(I - Q_t Q_t^T\right) \left(x_{t+1} x_{t+1}^* + \overline{x_{t+1}} x_{t+1}^T\right) \left(I - Q_t Q_t^T\right).$$

Therefore, if we can restrict the search space for the target eigenvector of (4.16) to $\mathbb{P}_t^s$, Lyapunov inverse iteration can be applied to (4.16) to compute $-Re(\mu_{t+1})$. Moreover, the analysis of section 3.4 (which applies to (4.2)) can be generalized directly to (4.16). This means that to compute $-Re(\mu_{t+1})$, it suffices to find an accurate solution to

$$\widehat{S}\mathcal{Y}_1 + \mathcal{Y}_1 \widehat{S}^T = -2\widehat{S} Z^{(1)} \widehat{S}^T \tag{4.17}$$

in $\mathbb{P}_t^s$. In general, solutions to (4.17) are not unique: any matrix of the form $\mathcal{Y}_1 + Q_t X Q_t^T$ where $X \in \mathbb{C}^{n\times n}$ is also a solution, since $\widehat{S} Q_t = \mathbf{0}$ by Lemma 4.7. However, in the designated search space $\mathbb{P}_t^s$, the solution to (4.17) is indeed unique. In addition, we can obtain estimates for the eigenpair $(\mu_{t+1}, \widehat{x}_{t+1})$ of $\widehat{S}$ in the same way we compute estimates for (μ_1, x_1) in section 2 (see (4.5)).

The analysis above leads to Algorithm 10 for computing k rightmost eigenvalues of $\mathbf{A}x = \mu \mathbf{M}x$. At each iteration of this algorithm, we compute the $(t+1)^{st}$ rightmost eigenvalue μ_{t+1}, or the $(t+1)^{st}$ and $(t+2)^{nd}$ rightmost eigenvalues $(\mu_{t+1}, \overline{\mu_{t+1}})$. The iteration terminates when k rightmost eigenvalues have been found. In this algorithm, we need to compute the eigenvalue with smallest modulus for several Lyapunov eigenvalue problems (4.16) corresponding to different values of t. One way to do this is to simply apply Algorithm 9 to each of these problems. In the

Algorithm 10: Compute k rightmost eigenvalues of $\mathbf{A}x = \mu\mathbf{M}x$

1. Initialization: $t = 0$, $\mathcal{E}_t = \emptyset$, $\mathcal{X}_t = \emptyset$, $Q_t = \mathbf{0}$, and $\widehat{S} = \left(I - Q_t Q_t^T\right) S$.
2. While $t < k$:
   - 2.1. Solve (4.16) for the eigenvalue with smallest modulus, $-Re(\mu_{t+1})$, and its corresponding eigenvector in $\mathbb{P}_t^s$.
   - 2.2. Compute an estimate $\left(\mu_{t+1}^{approx}, \widehat{x}_{t+1}^{approx}\right)$ for $(\mu_{t+1}, \widehat{x}_{t+1})$.
   - 2.3. Update:
     - if μ_{t+1}^{approx} is real:
       $\mathcal{E}_{t+1} \leftarrow \left\{\mathcal{E}_t, \mu_{t+1}^{approx}\right\}$, $\widehat{\mathcal{X}}_{t+1} \leftarrow \left[\widehat{\mathcal{X}}_t, \widehat{x}_{t+1}^{approx}\right]$, $t \leftarrow t + 1$;
     - else:
       $\mathcal{E}_{t+2} \leftarrow \left\{\mathcal{E}_t, \mu_{t+1}^{approx}, conj\left(\mu_{t+1}^{approx}\right)\right\}$,
       $\widehat{\mathcal{X}}_{t+2} \leftarrow \left[\widehat{\mathcal{X}}_t, \widehat{x}_{t+1}^{approx}, conj\left(\widehat{x}_{t+1}^{approx}\right)\right]$, $t \leftarrow t + 2$.
   - 2.4. Compute the thin QR factorization of $\widehat{\mathcal{X}}_t$: $[Q, R] = qr\left(\widehat{\mathcal{X}}_t, 0\right)$, and let $Q_t = Q$, $\widehat{S} = \left(I - Q_t Q_t^T\right) S$.

next section, we will discuss the way step 2.1 of Algorithm 10 is implemented, which is much more efficient. Note that the technique for computing $-Re(\mu_{t+1})$ $(t > 0)$ introduced in this section is based on the assumption that Q_t, whose columns form an orthonormal basis for $\{x_j\}_{j=1}^t$, is given. In Algorithm 10, Q_t is taken to be a matrix whose columns form an orthonormal basis for the columns of $\widehat{\mathcal{X}}_t$. Such an approach is justified in the next section as well.

We apply Algorithm 10 to compute a few rightmost eigenvalues for some cases of the examples considered in section 4.2. The results for step 2.1 in each iteration of Algorithm 10 are reported in Table 4.9 (see Table 4.8 for notation). For example, consider the driven-cavity flow at $Re = 7800$. From Table 4.9, we can find the eight rightmost eigenvalues of $\mathbf{A}x = \mu\mathbf{M}x$: $\mu_{1,2} = -0.00514 \pm 2.69845i$, $\mu_3 = -0.00845$, $\mu_{4,5} = -0.01531 \pm 0.91937i$, $\mu_{6,7} = -0.02163 \pm 1.78863i$, and $\mu_8 = -0.02996$ (see Figure 4.2(a)).

Table 4.8: Notation for Algorithm 10

Symbol	Definition
$\lambda^{(t)}$, $Z^{(t)}$	the estimated eigenvalue of (4.16) with smallest modulus ($-Re(\mu_{t+1})$) and its associated eigenvector, respectively
μ_{t+1}^{approx}	the estimated $(t+1)^{st}$ rightmost eigenvalue of $\mathbf{A}x = \mu\mathbf{M}x$, *i.e.*, μ_{t+1}
R_t^{eig}	$\widehat{S}Z^{(t)} + Z^{(t)}\widehat{S}^T + \lambda^{(t)}2\left(\widehat{S}Z^{(t)}\widehat{S}^T\right)$, *i.e.*, the residual of (4.17)

4.4 Implementation Details of Algorithm 10

In the previous section, we proposed an algorithm that finds k rightmost eigenvalues of (2.2) by computing the eigenvalue with smallest modulus for a series of Lyapunov eigenvalue problems (4.16) corresponding to different values of t. In this section, more details of how to implement this algorithm efficiently will be discussed.

4.4.1 Efficient Solution of the Lyapunov Eigenvalue Problems

We first make a preliminary observation.

Proposition 4.11. *The unique solution to (4.17) in $\mathbb{P}_t^s$ is*

$$\mathcal{Y}_1 = \left(I - Q_t Q_t^T\right) Y_1 \left(I - Q_t Q_t^T\right),$$

where Y_1 is the solution to (4.8).

Proof. The proof is straightforward with the help of Lemma 4.6. □

By Proposition 4.11, if we know the solution Y_1 to (4.8), we can formally write down the unique solution $\mathcal{Y}_1$ to (4.17) in $\mathbb{P}_t^s$. In practice, we do not know Y_1; instead, we solve (4.8) using an iterative solver (such as RKSM), which produces an approximate solution Y_1^{approx}. The following proposition shows that we can obtain

from Y_1^{approx} an approximate solution to (4.17) that is essentially as accurate a solution to (4.17) as Y_1^{approx} is to (4.8).

Proposition 4.12. *Let* $\mathcal{Y}_1^{approx} = \left(I - Q_t Q_t^T\right) Y_1^{approx} \left(I - Q_t Q_t^T\right)$ *and*

$$\mathcal{R}_1^{lyap} = \widehat{S}\mathcal{Y}_1^{approx} + \mathcal{Y}_1^{approx}\widehat{S}^T + 2\widehat{S}Z^{(1)}\widehat{S}^T.$$

Then $\|\mathcal{R}_1^{lyap}\|_F \leq 4\|R_1^{lyap}\|_F$, *where* $R_1^{lyap} = SY_1^{approx} + Y_1^{approx}S^T + 2SZ^{(1)}S^T$.

Proof. Using Lemma 4.6, we can show easily that

$$\mathcal{R}_1^{lyap} = \left(I - Q_t Q_t^T\right) R_1^{lyap} \left(I - Q_t Q_t^T\right).$$

Therefore,

$$\|\mathcal{R}_1^{lyap}\|_F \leq \left\|I - Q_t Q_t^T\right\|_F^2 \|R_1^{lyap}\|_F \leq \left(1 + \|Q_t\|_F^2\right)^2 \|R_1^{lyap}\|_F = 4\|R_1^{lyap}\|_F.$$

□

This analysis suggests the following strategy for step 2.1 of Algorithm 10: when $t = 0$, we compute $-Re(\mu_1)$ by applying Algorithm 9 to (4.2), in which a good approximate solution Y_1^{approx} to (4.8) is computed; in any subsequent iteration where $0 < t \leq k - 1$, instead of applying Lyapunov inverse iteration again to (4.16), we simply get the approximate solution $\mathcal{Y}_1^{approx}$ to (4.17) specified in Proposition 4.12, from which $-Re(\mu_{t+1})$ can be computed. Details of this approach are described below.

Recall that Y_1^{approx} computed by an iterative solver such as RKSM is of the form $\mathbf{V}_2 D_2 \mathbf{V}_2^T$, where $\mathbf{V}_2 \in \mathbb{R}^{n \times d_1}$ is orthonormal and $d_1 = rank(\mathbf{V}_2) \ll n$. We first rewrite $\mathcal{Y}_1^{approx}$ in a similar form $\mathbf{U}(\Sigma W^T D_2 W \Sigma)\mathbf{U}^T$, where $\mathbf{U}\Sigma W^T$ is the 'thin' singular value decomposition (SVD) of $\left(I - Q_t Q_t^T\right)\mathbf{V}_2$. Then we can compute an estimate for $-Re(\mu_{t+1})$ in the same way we compute estimates for $-Re(\mu_1)$ in Algorithms 8 or 9. That is, we solve the small, projected Lyapunov eigenvalue problem

$$\widetilde{S}_h \widetilde{Z} + \widetilde{Z}\widetilde{S}_h^T + \widetilde{\lambda}\left(2\widetilde{S}_h \widetilde{Z}\widetilde{S}_h^T\right) = 0 \tag{4.18}$$

for its eigenvalue with smallest modulus, where $\widetilde{S}_h = \mathbf{U}^T \widehat{S}\mathbf{U}$. Recall that the matrix $\widetilde{S} = \mathbf{V}_2^T S \mathbf{V}_2$ in (4.3) can be obtained with no additional cost from the Arnoldi decomposition (for instance, (3.7)). Here, $\widetilde{S}_h$ can be computed cheaply as well since

$$\mathbf{U}^T \widehat{S}\mathbf{U} = \Sigma^{-1} W^T {\mathbf{V}_2}^T \left(I - Q_t Q_t^T\right) S \mathbf{V}_2 W \Sigma^{-1}$$

by Lemma 4.6, and $S\mathbf{V}_2$ is given by the same Arnoldi decomposition. Let $\widetilde{\lambda}_1$ be the eigenvalue with smallest modulus of (4.18), and $\widetilde{Z}_1 = \widetilde{\mathcal{V}}\widetilde{\mathcal{D}}\widetilde{\mathcal{V}}^T$ be the eigenvector associated with $\widetilde{\lambda}_1$. Then the estimated $-Re(\mu_{t+1})$ and eigenvector of (4.16) associated with it are $\lambda^{(t)} = \widetilde{\lambda}_1$ and $Z^{(t)} = \mathcal{V}_t \widetilde{\mathcal{D}} \mathcal{V}_t^T$, where $\mathcal{V}_t = \mathbf{U}\widetilde{\mathcal{V}}$. In addition, by solving the small eigenvalue problem $\left(\mathcal{V}_t^T \widehat{S}\mathcal{V}_t\right) y = \theta y$, we get an estimate $\mu_{t+1}^{approx} = \frac{1}{\theta}$ for μ_{t+1} and $\widehat{x}_{t+1}^{approx} = \mathcal{V}_t y$ for $\widehat{x}_{t+1}$.

4.4.2 Efficient Computation of the Matrix Q_t

At each iteration of Algorithm 10, we need an orthornormal basis for $\{x_j\}_{j=1}^t$, the eigenvectors associated with the t rightmost eigenvalues of $\mathbf{A}x = \mu\mathbf{M}x$. When $t = 0$, we can get estimates for x_1 from Lyapunov inverse iteration applied to (4.16); however, when $t > 0$, we are only able to get estimates for the eigenvectors of the deflated matrix $\widehat{S}$. We will discuss how an orthonormal basis for $\{x_j\}_{j=1}^t$ can be computed efficiently from these estimates.

We first consider the simplest case where all k rightmost eigenvalues of $\mathbf{A}x = \mu\mathbf{M}x$ are real. In this case, we have the following result.

Proposition 4.13. *For any t such that $1 \leq t \leq k$,*

$$span\left\{\left(I - Q_{j-1}Q_{j-1}^T\right)x_j\right\}_{j=1}^t = span\left\{x_j\right\}_{j=1}^t. \tag{4.19}$$

Proof. We argue by induction.

1. When $t = 1$, since $Q_0 = \mathbf{0}$,

$$span\left\{\left(I - Q_0Q_0^T\right)x_1\right\} = span\left\{x_1\right\}.$$

The claim is trivially true.

2. When $t = 2$, since $x_2 = \left(I - Q_1 Q_1^T\right) x_2 + Q_1 \alpha_1$ where $\alpha_1 \in \mathbb{C}$,

$$\begin{aligned} span\left\{\left(I - Q_0 Q_0^T\right) x_1, \left(I - Q_1 Q_1^T\right) x_2\right\} &= span\left\{x_1, x_2 - Q_1 \alpha_1\right\} \\ &= span\left\{x_1, x_2\right\}. \end{aligned}$$

3. Assume the claim is true for any t that satisfies $3 \leq t \leq k-1$. Now we want to show that it is true for $t+1$. Note that $x_{t+1} = \left(I - Q_t Q_t^T\right) x_{t+1} + Q_t \alpha_t$, where $\alpha_t \in \mathbb{C}^t$. Then by the induction hypothesis,

$$\begin{aligned} span\left\{\left(I - Q_{j-1} Q_{j-1}^T\right) x_j\right\}_{j=1}^{t+1} &= span\left\{x_1, x_2, \ldots, x_t, x_{t+1} - Q_t \alpha_t\right\} \\ &= span\left\{x_j\right\}_{j=1}^{t+1}. \end{aligned}$$

□

Consequently, if we can find an orthonormal basis for $\left\{\left(I - Q_{j-1} Q_{j-1}^T\right) x_j\right\}_{j=1}^{t}$, then this is an orthonormal basis for $\{x_j\}_{j=1}^{t}$ as well. In Algorithm 10, the j^{th} column of the matrix $\widehat{\mathcal{X}}_t$ is approximately $\left(I - Q_{j-1} Q_{j-1}^T\right) x_j$; therefore, Q_t can be approximated by computing the thin QR factorization of $\widehat{\mathcal{X}}_t$.

As seen in Table 4.9, some of the k rightmost eigenvalues of $\mathbf{A}x = \mu \mathbf{M}x$ are not real. In this case, t may increase by 2 instead of 1 from one iteration to the next. Let $\mathcal{T} = \{t_i\}_{i=0}^{s}$ $(t_i < t_{i+1})$ be the collection of every value of t for which we need to solve the Lyapunov eigenvalue problem (4.16). Then $t_0 = 0$, $t_s = k-1$ or $k-2$, and $t_{i+1} - t_i = 1$ or 2. For example, in the case of the cavity flow at $Re = 7800$, if $k = 8$, then $\mathcal{T} = \{0, 2, 3, 5, 7\}$. In the same way we prove Proposition 4.19, we can

show that for any $1 \leq i \leq s$,

$$span\left\{\left(I - Q_{t_{j-1}} Q_{t_{j-1}}^T\right) x_{t_j}, \left(I - Q_{t_j} Q_{t_j}^T\right) \overline{x_{t_j}}\right\}_{j=1}^{i} = span\left\{x_{t_j}, \overline{x_{t_j}}\right\}_{j=1}^{i}. \quad (4.20)$$

In Algorithm 10, if μ_{t_j} is real, then the ${t_j}^{th}$ column of $\widehat{\mathcal{X}}_{t_i}$ is approximately

$$\left(I - Q_{t_{j-1}} Q_{t_{j-1}}^T\right) x_{t_j};$$

otherwise, the ${t_j}^{th}$ and $(t_j - 1)^{st}$ columns of $\widehat{\mathcal{X}}_{t_i}$ hold estimates for

$$\left(I - Q_{t_{j-1}} Q_{t_{j-1}}^T\right) x_{t_j} \text{ and } \left(I - Q_{t_j} Q_{t_j}^T\right) \overline{x_{t_j}}.$$

By (4.20), Q_{t_i} can be approximated by computing the thin QR factorization of $\widehat{\mathcal{X}}_{t_i}$.

4.5 Conclusions

In this chapter, we have developed a robust and efficient method of computing a few rightmost eigenvalues of (2.2) at any point $(\overline{u}_0, \alpha_0)$ in the stable regime. We have shown that the distance between the rightmost eigenvalue of (2.2) and the imaginary axis is the eigenvalue with smallest modulus of an $n^2 \times n^2$ eigenvalue problem (4.1). Since (4.1) has the same Kronecker structure as the one considered in [29] and Chapter 3, this distance can be computed by the Lyapunov inverse iteration developed and studied in these references, which also produces estimates of the rightmost eigenvalue as by-products. An analysis of the fast convergence of

Lyapunov inverse iteration in this particular application is given, which indicates that the algorithm will converge in two steps as long as the first Lyapunov equation is solved accurately enough. Furthermore, assuming t rightmost eigenpairs are known, we show that all the main theoretical results proven for (4.1) can be generalized to the deflated problem (4.15), whose eigenvalue with smallest modulus is the distance between the $(t+1)^{st}$ rightmost eigenvalue and the imaginary axis. Finally, an algorithm that computes a few rightmost eigenvalues of (2.2) is proposed, followed by a discussion on how to implement it efficiently. The method developed in this study together with the method proposed in [29] and Chapter 3 constitute a robust way of detecting the transition to instability in the steady-state solution of a large-scale dynamical system.

Table 4.9: Algorithm 10 applied to Examples 1, 2 and 3

t	$\lambda^{(t)}$	μ_{t+1}^{approx}	$\|R_t^{eig}\|_F$
Example 1 (Re=6000), $k = 8$			
0	0.01084	-0.01084	7.11628e-10
1	0.02006	-0.02006+0.91945i	5.31308e-11
3	0.03033	-0.03033+1.79660i	1.57820e-11
5	0.03794	-0.03794	2.27041e-10
6	0.04418	-0.04418+2.69609i	4.50346e-11
Example 1 (Re=7800), $k = 8$			
0	0.00514	-0.00514+2.69845i	3.62567e-11
2	0.00845	-0.00845	1.92675e-09
3	0.01531	-0.01531+0.91937i	1.06201e-10
5	0.02163	-0.02163+1.78863i	6.81321e-11
7	0.02996	-0.02996	1.86935e-10
Example 2 (Re=300), $k = 6$			
0	0.10405	-0.10405+2.22643i	6.10287e-09
2	0.32397	-0.32397	2.83185e-10
3	0.39197	-0.39197	3.34178e-11
4	0.60628	-0.60628	1.31394e-07
5	0.87203	-0.87203	1.77364e-06
Example 2 (Re=350), $k = 6$			
0	0.02411	-0.02411+2.24736i	7.16343e-09
2	0.28408	-0.28408	1.46365e-10
3	0.33571	-0.33571	3.81057e-11
4	0.56485	-0.56485	6.03926e-08
5	0.79196	-0.79196	9.20079e-07
Example 3, $k = 6$			
0	0.05000	-0.05000+25.0000i	6.88230e-09
2	0.10000	-0.10000	1.50946e-12
3	0.20000	-0.20000	8.60934e-09
4	0.30000	-0.30000	1.31349e-07
5	0.40000	-0.40000	4.08853e-06

Chapter 5

Efficient Iterative Solution of the Linear Systems Arising from Lyapuonv Inverse Iteration

In Chapters 3 and 4, we have shown two ways that Lyapunov inverse iteration can be utilized in linear stability analysis. First, at a stable point $(\overline{u}, \alpha_0)$ close to the critical point $(\overline{u}_c, \alpha_c)$, it can be applied to compute $\lambda_c = \alpha_c - \alpha_0$, the eigenvalue of (3.14) with smallest modulus. The critical parameter value α_c can then be obtained easily as $\alpha_0 + \lambda_c$. Second, at any stable point, Lyapunov inverse iteration can be used for computing the distance $-Re(\mu_1)$ between the rightmost eigenvalue μ_1 of (2.2) and the imaginary axis, which is also the eigenvalue of (4.2) with smallest modulus. A few rightmost eigenvalues of (2.2) can be obtained from the computation of $-Re(\mu_1)$ for almost no additional cost.

The two eigenvalue problems (3.14) and (4.2) are very similar in structure and applying Lyapunov inverse iteration to them requires iterative solution of a large-scale Lyapunov equation in the form of

$$SY + YS^T = PCP^T \tag{5.1}$$

at each step. In (5.1), $S = \mathbf{A}^{-1}\mathbf{M} \in \mathbb{R}^{n\times n}$ and its eigenvalues all lie in the left half of the complex plane; $P \in \mathbb{R}^{n\times p}$ is an orthonormal matrix with rank $p = 1, 2$

or 4 (see Algorithms 7 and 9). Consequently, the implementation of Lyapunov inverse iteration depends on solving (5.1) efficiently. Iterative solution of (5.1) (see, for instance, [12, 39, 42]) entails matrix-vector products with S and/or $(S - sI)^{-1}$ where $s \in \mathbb{C}$ is a shift. In Chapters 3 and 4, different iterative Lyapunov solvers were tested and compared in solving (5.1). However, the study in previous chapters is incomplete since the linear systems were solved using a sparse direct method that requires factorization of the coefficient matrices; in practice, due to the dimension of the problem, they must be solved by an iterative linear solver in conjunction with a proper preconditioning strategy. The aim of this chapter is to complete the discussion on Lyapunov solvers by implementing iterative solution methods for the linear systems arising from them and more importantly, to explore means of reducing the cost of these iterative solves.

This chapter is organized as follows. In section 5.1, we review the iterative Lyapunov solvers considered in Chapters 3 and 4 and introduce the types of linear systems that need to be solved. In Section 5.2, we first review and test the iterative methods developed for such systems arising from incompressible Navier-Stokes equations. Then we incorporate these methods into the Lyapunov solvers, which are tested on several examples considered in previous chapters. Based on the numerical results, we propose in section 5.3 a modified version of the rational Krylov subspace method that is able to achieve more than 50% savings in the computational cost. Some concluding remarks are given in section 5.4.

5.1 Review of Iterative Lyapunov Solvers

In this section, we review the Lyapunov solvers used for Lyapunov inverse iteration considered in Chapters 3 and 4: the standard Krylov subspace method [24, 39] and the rational Krylov subspace method (RKSM) [10, 12]. Both methods seek an approximate solution to (5.1) of the form $\mathbf{V}Q\mathbf{V}^T$, where $\mathbf{V}$ is an orthonormal matrix whose columns span a small subspace of $\mathbb{R}^n$, and Q is the solution to a small Lyapunov equation that can be solved using direct methods. This small Lyapunov equation is obtained by imposing the Galerkin condition (3.4) on the residual

$$R = S\left(\mathbf{V}Q\mathbf{V}^T\right) + \left(\mathbf{V}Q\mathbf{V}^T\right)S^T - PCP^T.$$

Our emphasis here is on describing the types of linear systems that will arise from the two Lyapunov solvers when applied to (5.1).

In the standard Krylov method, the Krylov subspace that we are familiar with,

$$\mathcal{K}_m(S,P) = \left\{P, SP, S^2P, \ldots, S^{m-1}P\right\},$$

is built. We restate below an algorithm that implements this method, which is already given by Algorithm 5 in Chapter 3.

In Algorithm 11, the matrix $H_m = \mathbf{V}^T S\mathbf{V}$ can be obtained for no additional cost since it is simply the $mp \times mp$ leading principal minor $[H_{i,j}]_{i,j=1}^{m}$ of the matrix H, where p is the rank of P. As shown in section 3.1, the residual norm $\|R\|_F$ of (5.1) can be computed cheaply as well. At each step of Algorithm 11, the computation

Algorithm 11: The standard Krylov method for (5.1)

1. Given a tolerance τ. Let $V_1 = \mathbf{V} = P$.
2. For $m = 1, 2, \cdots$
   - 2.1. $W = SV_m$.
     for $i = 1, \ldots, m$
     $H_{i,m} \leftarrow V_i^T W$;
     $W \leftarrow W - V_i H_{i,m}$.
   - 2.2. Solve the small Lyapunov equation
     $$H_m Q + Q H_m^T = \left(\mathbf{V}^T P\right) C \left(\mathbf{V}^T P\right)^T$$
     where $H_m = \mathbf{V}^T S \mathbf{V}$.
   - 2.3. Compute the reduced QR factorization of W: $W = V_{m+1} H_{m+1,m}$.
   - 2.4. If the residual norm $\|R\|_F < \tau$, then stop.
   - 2.5. Else, $\mathbf{V} \leftarrow [\mathbf{V}, V_{m+1}]$.

of p matrix-vector products SV_m is required. Since $S = \mathbf{A}^{-1}\mathbf{M}$ in (5.1), the p matrix-vector products entail p solves of the linear system

$$\mathbf{A}x = b, \tag{5.2}$$

precisely the kind that needs to be solved in the computation of the steady-state solution for (2.1).

The rational Krylov subspace method was originally developed in [36, 37] for the computation of the interior eigenvalues of (2.2). This method constructs the subspace

$$\mathcal{K}_m(S, P, \mathbf{s}) = \left\{ P, (S - s_1 I)^{-1} P, (S - s_2 I)^{-1}(S - s_1 I)^{-1} P, \ldots, \prod_{j=1}^{m-1} (S - s_{m-j} I)^{-1} P \right\},$$

where $\mathbf{s} = \{s_j\}_{j=1}^{m-1} \in \mathbb{C}^{m-1}$ is a set of shifts that need to be selected by some means. Recently, its utility in solving large-scale Lyapunov equations has been investigated

in [12]. An algorithm of this method applied to (5.1) reads as Algorithm 12.

Algorithm 12: The rational Krylov subspace method for (5.1)

1. Given a tolerance τ and a shift s_1. Let $V_1 = \mathbf{V} = P$.
2. For $m = 1, 2, \cdots$
   - 2.1. $W = (S - s_m I)^{-1} V_m$.
     for $i = 1, \ldots, m$
     $H_{i,m} \leftarrow V_i^T W$;
     $W \leftarrow W - V_i H_{i,m}$.
   - 2.2. Compute the reduced QR factorization of W: $W = V_{m+1} H_{m+1,m}$.
   - 2.3. Compute $T_m = \mathbf{V}^T S \mathbf{V}$ and solve the small Lyapunov equation

$$T_m Q + Q T_m^T = \left(\mathbf{V}^T P\right) C \left(\mathbf{V}^T P\right)^T.$$

   - 2.4. If $\|R\|_F < \tau$, then stop.
   - 2.5. Else, $\mathbf{V} \leftarrow [\mathbf{V}, V_{m+1}]$ and compute the next shift s_{m+1}.

In [11] (for symmetric S) and [12] (for general S), adaptive and parameter-free approaches for generating the shifts $\mathbf{s}$ were proposed. In [12], the first shift s_1 is chosen to be a rough estimate of either $-Re_{min}(\theta)$ or $-Re_{max}(\theta)$, where $Re_{min}(\theta)$ and $Re_{max}(\theta)$ denote the minimum and maximum real parts of the eigenvalues of S, respectively. (Since the eigenvalues of S all lie in the left half of the complex plane, $0 > Re_{max}(\theta) > Re_{min}(\theta)$.) Let $\mathcal{I} = [-Re_{max}(\theta), -Re_{min}(\theta)]$ and $\left\{\widehat{\theta}_j\right\}_{j=1}^{mp}$ denote the eigenvalues of T_m, where p is the rank of P. Each subsequent shift s_{m+1} is then chosen as follows:

$$s_{m+1} = \arg\left(\max_{s\in\mathcal{I}} \frac{1}{|r_m(s)|}\right), \text{ where } r_m(s) = \frac{\prod_{j=1}^{mp}\left(s - \widehat{\theta}_j\right)}{\prod_{j=1}^{m}(s - s_j)^p}. \tag{5.3}$$

Once $\left\{\widehat{\theta}_j\right\}_{j=1}^{mp}$ are known, this selection process only involves evaluating the rational function $r_m(s)$ at some sample points in $\mathcal{I}$, which is cheap.

Consider the linear solves required by Algorithm 12 when it is applied to (5.1).

Since

$$(S - s_m I)^{-1} = \left(\mathbf{A}^{-1}\mathbf{M} - s_m \mathbf{A}^{-1}\mathbf{A}\right)^{-1} = \left(\mathbf{M} - s_m \mathbf{A}\right)^{-1} \mathbf{A}, \tag{5.4}$$

the computation of $(S - s_m I)^{-1} V_m$ at step 2.1 of Algorithm 12 entails solving p linear systems of the form

$$(\mathbf{M} - s\mathbf{A})x = b \tag{5.5}$$

with $s > 0$. Note that the structure of (5.5) is exactly like that of the linear systems that need to be solved in the computation of the time-dependent solutions of (2.1), where s plays the role of the time step Δt.

Unlike in Algorithm 11, extra work is needed in Algorithm 12 to obtain the matrix $T_m = \mathbf{V}^T S \mathbf{V}$. Computing it naively requires mp matrix-vector products with S, which entail mp solves with $\mathbf{A}$ since $S = \mathbf{A}^{-1}\mathbf{M}$ in (5.1). A more efficient way of computing this matrix was proposed in [37] (see also Proposition 4.1 of [12]) that only requires knowing SV_{m+1}, or equivalently, p solves of (5.2). The matrix SV_{m+1} is also needed for efficient computation of the residual norm $\|R\|_F$ (see Proposition 4.2 of [12]).

To sum up, each iteration of the standard Krylov subspace method (Algorithm 11) applied to (5.1) requires p solves of (5.2) to expand the Krylov subspace; each iteration of RKSM requires p solves of (5.5) to expand the Krylov subspace and another p solves of (5.2) in order to compute the projection T_m of S onto the Krylov subspace.

5.2 Numerical Results

In the previous section, we have shown that solving the Lyapunov equation (5.1) arising from Lyapunov inverse iteration requires multiple solves of the linear systems (5.2) and/or (5.5), which are essentially the types of equations that arise in solving steady or transient PDEs. Therefore, iterative solution methods developed for these problems can be applied directly in Lyapunov inverse iteration. Such strategies frequently involve designing efficient preconditioners that take advantage of the characteristics of the underlying differential operator.

The dynamical system (2.1) that we are going to consider in this section is again spatial discretization of the Navier-Stokes equations modeling incompressible flows, namely, (3.1). In order to examine the performance of Lyapunov solvers with iterative linear solves, we proceed in two steps: first, we review the preconditioners developed for (3.1) and test them on (5.2) and (5.5); then we integrate these solution techniques into the Lyapunov solvers described in the previous section and apply them to (5.1).

5.2.1 Iterative Solves of the Linear Systems Arising from Lyapunov Inverse Iteration

The matrices $\mathbf{A}$ and $\mathbf{M}$ arising from div-stable mixed finite element discretization of (3.1) have the block structure

$$\mathbf{A} = \begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \text{ and } \mathbf{M} = \begin{bmatrix} -G & 0 \\ 0 & 0 \end{bmatrix}. \tag{5.6}$$

The matrix blocks $F \in \mathbb{R}^{n_u \times n_u}$, $B \in \mathbb{R}^{n_p \times n_u}$ and $G \in \mathbb{R}^{n_u \times n_u}$ are all sparse, where $n_u + n_p = n$ and $n_p < n_u$. G is the velocity mass matrix, B is the matrix representation of the discrete divergence operator, and F resembles the matrix representation of the discrete convection-diffusion operator. (The precise definition of F can be found in [13].) Since the theory of Lyapunov inverse iteration (see [29] and Chapter 4) requires a nonsingular mass matrix $\mathbf{M}$, as in Chapters 3 and 4, the mass matrix $\mathbf{M}$ is replaced by

$$\begin{bmatrix} -G & \eta B^T \\ \eta B & 0 \end{bmatrix},$$

where η is again chosen to be -10^{-2}.

Suppose $\mathbf{P}$ is a right preconditioner of (5.2) and GMRES is used as the iterative linear solver for the preconditioned system $(\mathbf{A}\mathbf{P}^{-1})\, y = b$. We seek an approximate solution of this system in the Krylov subspace

$$\mathcal{K}_m\left(\mathbf{A}\mathbf{P}^{-1}, b\right) = \left\{b, \left(\mathbf{A}\mathbf{P}^{-1}\right) b, \left(\mathbf{A}\mathbf{P}^{-1}\right)^2 b, \ldots, \left(\mathbf{A}\mathbf{P}^{-1}\right)^{m-1} b\right\}. \tag{5.7}$$

This means that at each GMRES step, we need to solve a linear system with the coefficient matrix $\mathbf{P}$. Once y has been computed, the solution x to (5.2) can be recovered as $\mathbf{P}^{-1}y$.

A good preconditioner $\mathbf{P}$ for (5.2) should satisfy the following two requirements: the eigenvalues of the matrix $\mathbf{A}\mathbf{P}^{-1}$ are tightly clustered, and the inverse of $\mathbf{P}$ can be efficiently applied to a vector. Note that $\mathbf{A}$ defined in (5.6) has the LU factorization

$$\begin{bmatrix} I & 0 \\ BF^{-1} & I \end{bmatrix} \begin{bmatrix} F & B^T \\ 0 & -BF^{-1}B^T \end{bmatrix} = \mathbf{L}\mathbf{U}.$$

If the preconditioner $\mathbf{P}$ is chosen to be $\mathbf{U}$, then all the eigenvalues of $\mathbf{AP}^{-1} = \mathbf{L}$ equal 1. This implies that if we apply GMRES to the preconditioned system, it will converge in just two steps. However, this choice of $\mathbf{P}$ is not practical since there is no good way of applying the inverse of the (dense) Schur complement $BF^{-1}B^T$. Preconditioners for (5.2) of the form

$$\mathbf{P} = \begin{bmatrix} P_F & B^T \\ 0 & -P_S \end{bmatrix} \tag{5.8}$$

were developed in [13, 14], where P_F is a preconditioner for F and P_S is a preconditioner for the Schur complement. In (5.8), we can choose P_F to be F itself and apply its inverse using a multigrid process (see [13]). The main issue in designing $\mathbf{P}$ is the choice of P_S.

In [13], two effective preconditioning strategies were proposed for the Schur complement: the pressure convection-diffusion preconditioner (PCD) and the least-squares commutator preconditioner (LSC). They are derived by minimizing the discrete version of a commutator of certain differential operators using two different heuristics (see [13]). The pressure convection-diffusion preconditioner is defined to be

$$A_p F_p^{-1} G_p$$

where A_p and F_p are the Laplacian matrix and convection-diffusion matrix in the pressure space, respectively, and G_p is the pressure mass matrix. The least-squares commutator preconditioner is defined to be

$$\left(B\widehat{G}^{-1}B^T\right)\left(B\widehat{G}^{-1}F\widehat{G}^{-1}B^T\right)^{-1}\left(B\widehat{G}^{-1}B^T\right)$$

where $\widehat{G} = \mathrm{diag}(G)$, the diagonal of the velocity mass matrix.

The coefficient matrix $\mathbf{M} - s\mathbf{A}$ of (5.5) has the block structure

$$\begin{bmatrix} -(sF+G) & (\eta - s)B^T \\ (\eta - s)B & 0 \end{bmatrix}$$

similar to that of $\mathbf{A}$, and the pressure convection-diffusion preconditioner and the least-squares commutator preconditioner can be derived in an analogous manner for its Schur complement $-(s-\eta)^2 B(sF+G)^{-1}B^T$. They are

$$-(s-\eta)^2 A_p(sF_p + G_p)^{-1}G_p$$

and

$$-(s-\eta)^2 \left(B\widehat{G}^{-1}B^T\right)\left(B\widehat{G}^{-1}(sF+G)\widehat{G}^{-1}B^T\right)^{-1}\left(B\widehat{G}^{-1}B^T\right),$$

respectively. The shift s in the rational Krylov subspace method is in general different from one iteration to another. An important feature of both preconditioners is that they do not require any extra work to construct as s varies.

The two preconditioners were improved in [15] by prescribing appropriate boundary conditions to the commutator that needs to be minimized in their derivation. It is demonstrated in [15] that with the modified preconditioners, the performance of GMRES is improved considerably.

Applying the inverse of the pressure diffusion-convection preconditioner to a vector requires a solve with A_p and a solve with G_p, *i.e.*, a Poisson solve and a solve with the pressure mass matrix; the application of the inverse of the least-squares commutator preconditioner, on the other hand, requires two solves with $B\widehat{G}^{-1}B^T$,

i.e., two Poisson solves (see [13]). Since these subsidiary problems can be solved efficiently using multigrid and the primary cost of the application of $\mathbf{P}^{-1}$ is the application of P_F^{-1}, the difference between the cost of applying the inverses of the two preconditioners for the Schur complement is not that important.

We investigate the utility of the preconditioning strategies proposed in [15] in solving (5.2) and (5.5). Recall from section 5.1 that we need to solve (5.2) with different right-hand sides in both Algorithms 11 and 12, and we need to solve (5.5) with different right-hand sides and different shifts in Algorithm 12. The performance of the solvers for (5.2) and (5.5) as the right-hand side and the shift vary will be reported in the next section. Here we simply take the right-hand side b of both (5.2) and (5.5) to be the unit vector whose entries all equal to $n^{-\frac{1}{2}}$ and consider five representative values of the shift, *i.e.*, $s = 10^{-3}, 10^{-2}, 10^{-1}, 1, 10$. The stopping criteria of GMRES are

$$\|\mathbf{A}x - b\|_2 < 10^{-10} \cdot \|b\|_2 \tag{5.9}$$

for (5.2) and

$$\|(\mathbf{M} - s\mathbf{A})\, x - b\|_2 < 10^{-10} \cdot \|b\|_2 \tag{5.10}$$

for (5.5). All the solves with P_F and P_S are done using a direct method, although in practice, they can be handled efficiently by a multigrid process.

The following four examples are considered in this section, where (3.1) are discretized using Q_2-Q_1 mixed finite element:

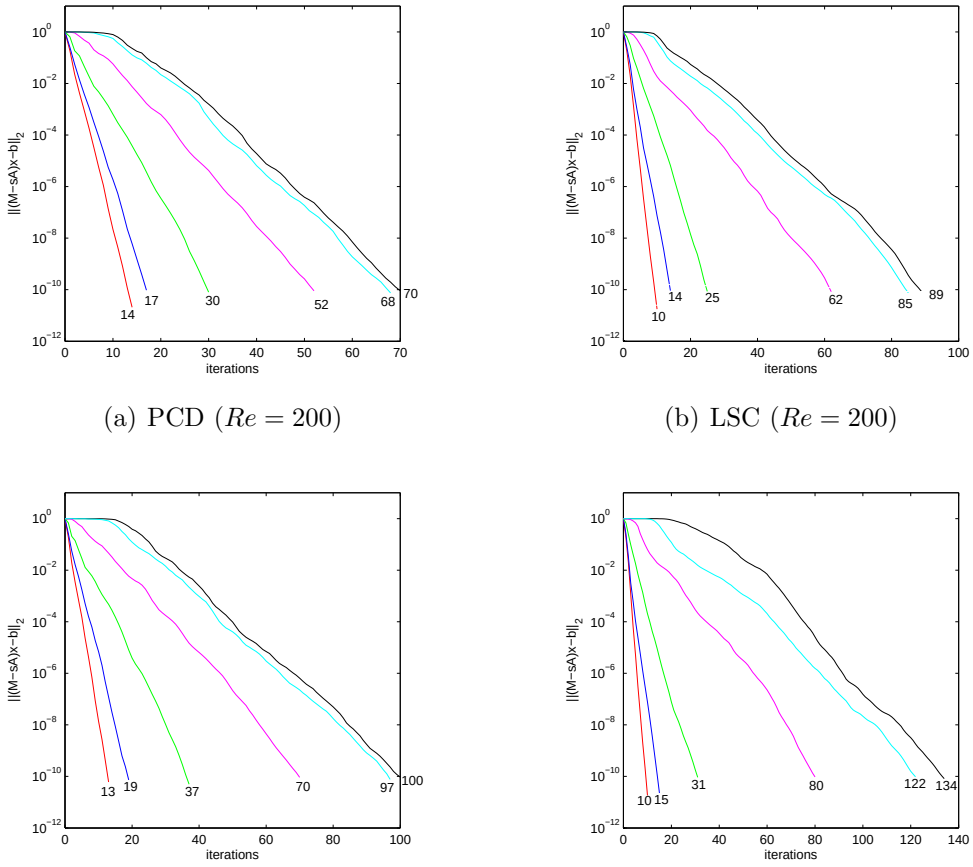
- flow over an obstacle (32×128 mesh, $n = 9512$) at $Re = 200$ and $Re = 350$;

and

- driven-cavity flow (64×64 mesh, $n = 9539$) at $Re = 2000$ and $Re = 4000$.

These two flows have been considered in previous chapters as well. Figure 5.1 displays the performance of the two preconditioners in solving (5.2) and (5.5) arising from the two examples of flow over an obstacle. The top two plots correspond to $Re = 200$ and the bottom two correspond to $Re = 350$. In each subplot, the decay of the residual norms $\|\mathbf{A}x - b\|_2$ and $\|(\mathbf{M} - s\mathbf{A})x - b\|_2$ is plotted against the number of preconditioned GMRES steps. From the left to the right, the six residual curves correspond to (5.5) with the five choices of s specified above and (5.2). The number next to each curve indicates the total number of GMRES steps required to meet the criterion (5.9) or (5.10). The performance of GMRES for these representative shifts shows that the smaller the shift s is, the easier it is to solve (5.5). This is because the mass matrix $\mathbf{M}$ will become more dominant in (5.5) as s get smaller, and a solve with $\mathbf{M}$ is much easier than a solve with the Jacobian matrix $\mathbf{A}$. The number of GMRES steps needed by (5.2) shows the limit of how expensive solving (5.5) can get as s increases. In addition, for both examples ($Re = 200$ and $Re = 350$), in the regime where the mass matrix $\mathbf{M}$ is dominant (*i.e.*, when $s < 1$), the least-squares commutator preconditioner outperforms the pressure convection-diffusion preconditioner; in the scenario where the Jacobian matrix $\mathbf{A}$ dominates, the pressure convection-diffusion preconditioner is more effective.

The performance of the least-squares commutator preconditioner and the pressure convection-diffusion preconditioner in solving (5.2) and (5.5) for the two exam-



(a) PCD ($Re = 200$)

(b) LSC ($Re = 200$)

(c) PCD ($Re = 350$)

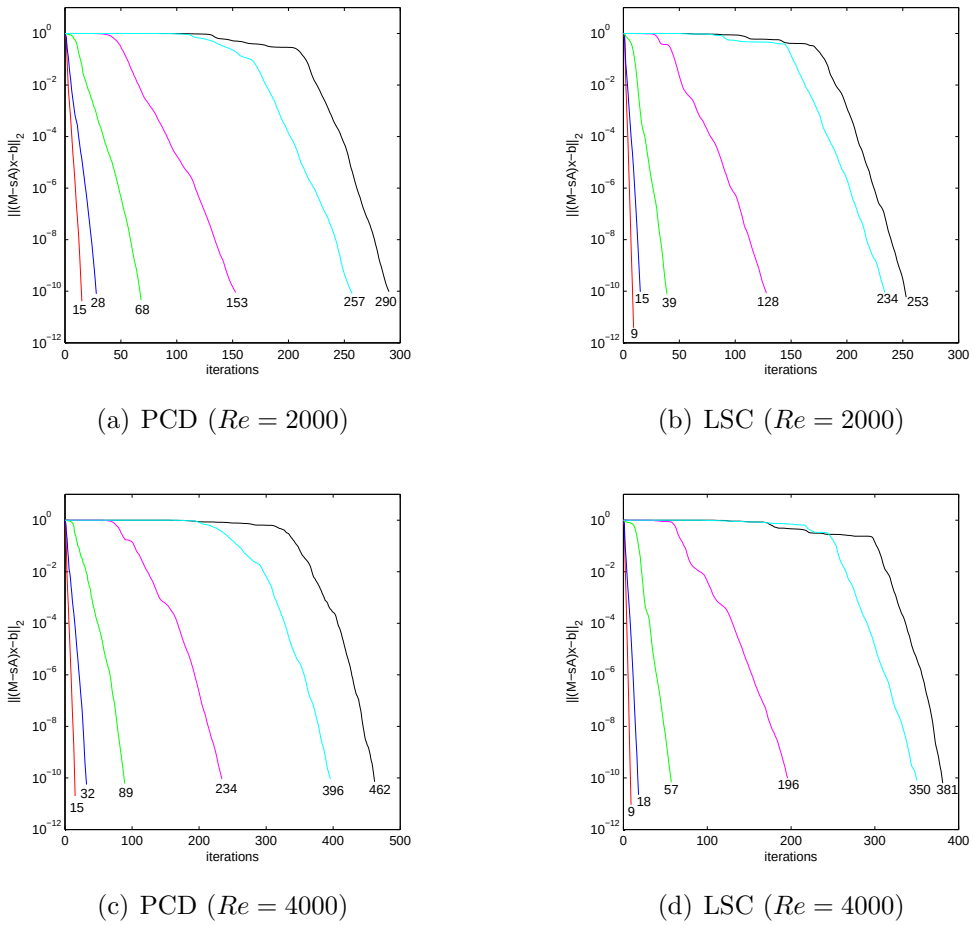
(d) LSC ($Re = 350$)

Figure 5.1: The performance of the pressure convection-diffusion (PCD) preconditioner and the least-squares commutator (LSC) preconditioner in flow over an obstacle

ples of driven-cavity flow is reported in Figure 5.2. The top two plots correspond to $Re = 2000$ and the bottom two correspond to $Re = 4000$. As observed for flow over an obstacle, (5.5) gets more difficult to solve as the shift s increases, with (5.2) as the limit. As can be seen in Figure 5.2, GMRES with the least-squares commutator preconditioner converges in fewer iterations than GMRES with the pressure convection-diffusion preconditioner for both Reynolds numbers and for both (5.2) and (5.5) under all choices of the shift. Interestingly, when $s = 1$, that is, when the Jacobian matrix $\mathbf{A}$ and the mass matrix $\mathbf{M}$ are equally weighted in (5.5), the

number of GMRES iterations needed is approximately half of what is needed for (5.2).

From Figures 5.1 and 5.2, we can also observe that as the Reynolds number grows, (5.2) and (5.5) with a larger shift become increasingly difficult to solve.



(a) PCD ($Re = 2000$)

(b) LSC ($Re = 2000$)

(c) PCD ($Re = 4000$)

(d) LSC ($Re = 4000$)

Figure 5.2: The performance of the pressure convection-diffusion (PCD) preconditioner and the least-squares commutator (LSC) preconditioner in driven-cavity flow

Recall from sections 3.3.1 and 3.3.2 that the critical Reynolds number of flow over an obstacle is about 370 and that of driven-cavity flow is about 8000. Thus, for flow over an obstacle, 200 is a medium Reynolds number and 350 is a high Reynolds number; for driven-cavity flow, on the other hand, 2000 is still considered to be

a low Reynolds number and 4000 a medium Reynolds number. We choose not to explore driven-cavity flow at high Reynolds numbers since the driven-cavity flow considered here is two-dimensional and it becomes a fictitious flow at high Reynolds numbers (see [41]). Experimental results [25, 26, 27] show that cavity flow at a high Reynolds number is neither two-dimensional nor steady. Therefore, the critical Reynolds number (about 8000) found in our numerical experiments (see section 3.3.1) is artificially high. In three-dimensional driven-cavity flow, this number is found to be approximately 2000 [33], which is significantly smaller. For the purpose of this chapter, the range of Reynolds numbers that we consider, $[200, 4000]$, is fairly representative.

5.2.2 Lyapunov Solvers with Iterative Linear Solves

In this section, we present numerical results of the Lyapunov solvers (with iterative linear solves) described in section 5.1. Recall that at any stable point $(\overline{u}_0, \alpha_0)$, the eigenvalue of (4.2) with smallest modulus is $-Re(\mu_1)$, the distance between the rightmost eigenvalue of (2.2) and the imaginary axis; if this point is also close to the critical point $(\overline{u}_c, \alpha_c)$, then the eigenvalue of (3.14) with smallest modulus is $\lambda_c = \alpha_c - \alpha_0$. Lyapunov equations of the form (5.1) arise from Lyapunov inverse iteration applied to both eigenvalue problems (3.14) and (4.2). Our main focus is (5.1) arising from Lyapunov inverse iteration applied to (4.2). Such a choice allows us to explore the behavior of the Lyapunov solvers for a broader range of problems since the eigenvalue of (4.2) with smallest modulus is well-defined at any stable

point, whereas that of (3.14) is only meaningful in the neighborhood of the critical point. For example, the eigenvalue of (3.14) with smallest modulus at $Re = 2000$ or $Re = 4000$ does not have any physical meaning for driven-cavity flow since they are too far away from the critical Reynolds number of this flow. Note that only one solve of (5.1) that has right-hand side of rank 1 is needed when we apply Algorithm 9 to (4.2) (see section 4.2.4). Unless otherwise stated, (5.1) refers to this Lyapunov equation.

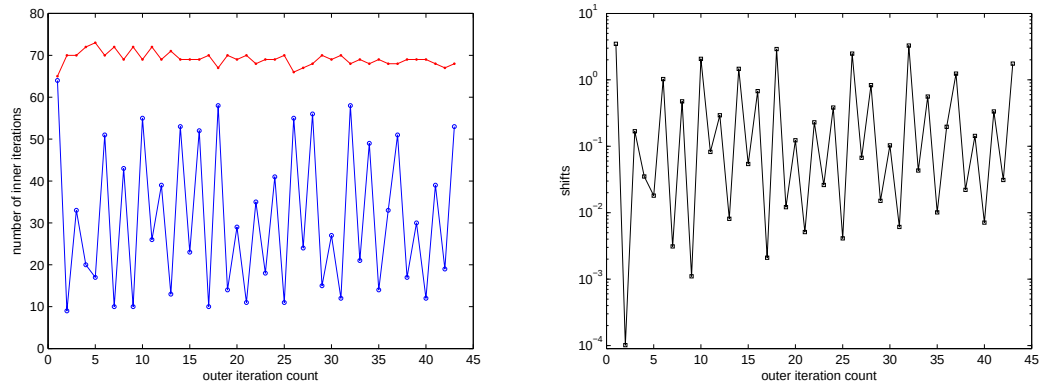
We again consider the following four examples: flow over an obstacle at $Re = 200$ and $Re = 350$ and driven-cavity flow at $Re = 2000$ and $Re = 4000$. The stopping criteria are

$$\|R\|_F < 10^{-6} \cdot \|C\|_F \tag{5.11}$$

for the Lyapunov solve (outer iteration) and (5.9), (5.10) for the linear solves (inner iteration). All the linear systems arising from the Lyapunov solvers will be solved using GMRES together with the preconditioners described in the previous section.

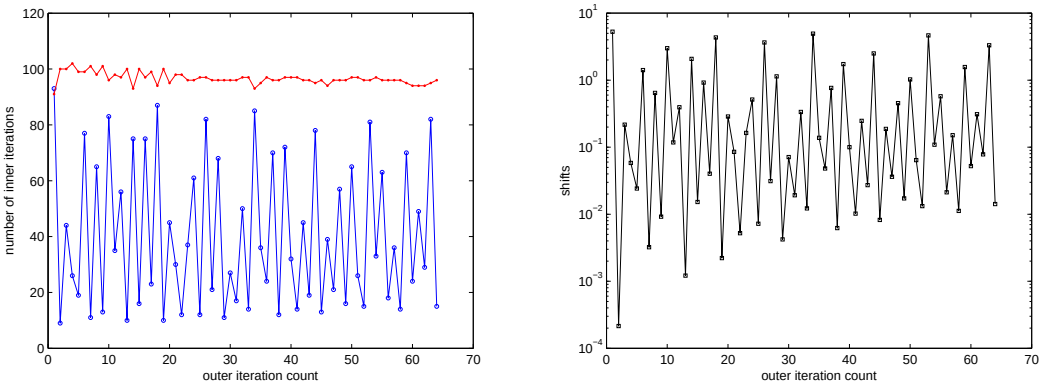
We first consider the rational Krylov subspace method (Algorithm 12). Since the right-hand side of (5.1) is rank-1, one solve of (5.2) and one solve of (5.5) are needed at each iteration of Algorithm 12. For the two examples of flow over an obstacle, the pressure convection-diffusion preconditioner is chosen for (5.5) with $s > 1$ and (5.2), whereas the least-square commutator preconditioner is used if $s < 1$. This choice is based on the numerical experiments in the previous section. Numerical results of Algorithm 12 are reported in Figures 5.3, in which we plot both the number of GMRES steps (inner iterations) and the shift s for each iteration of

this algorithm (outer iteration).



(a) iteration count ($Re = 200$)

(b) shifts ($Re = 200$)



(c) iteration count ($Re = 350$)

(d) shifts ($Re = 350$)

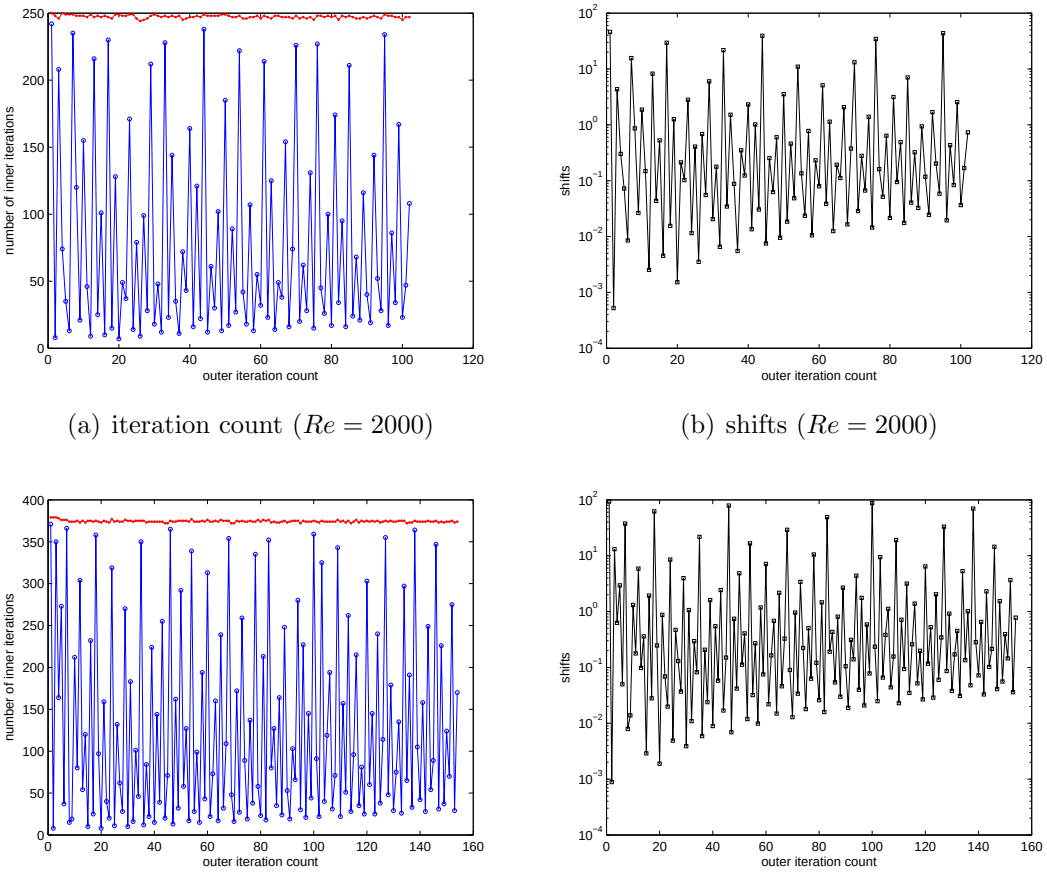
Figure 5.3: Algorithm 12 with an iterative linear solver applied to flow over an obstacle (•: the number of GMRES steps needed for solving (5.2), ∘: the number of GMRES steps needed for solving (5.5), □: the shift in (5.5))

In Figures 5.3(a) and 5.3(c), there are two curves representing respectively the number of GMRES steps needed for solving (5.2) (denoted by '•') and (5.5) (denoted by '∘') as Algorithm 12 proceeds. As can be seen from these two figures, the number of GMRES steps needed for solving (5.2) does not change much as the outer iteration advances. It is about 70 for $Re = 200$ and about 100 for $Re = 350$, as indicated by the rightmost curve in Figures 5.1(a) and 5.1(c). On the other hand, from the same pair of figures, we can see that the number of GMRES steps required

to sovle (5.5) is quite oscillatory and can change drastically from one iteration to the next. The pattern of oscillation of this number matches perfectly with that of the shift, which is depicted in Figures 5.3(b) and 5.3(d) (denoted by '□'). The bigger the shift is, the more GMRES steps are needed to solve (5.5). This behavior is again expected from the numerical experiments in the previous section. In many outer iterations, the number of GMRES steps required to solve (5.5) is much smaller than that needed by (5.2). On average, at $Re = 200$, it takes about 31 GMRES steps to solve (5.5) and as many as 69 GMRES steps to solve (5.2), and these two numbers become 40 and 97 for $Re = 350$. In other words, about 70% of all the GMRES steps taken to solve (5.1) are devoted to the solution of (5.2).

For the two examples of driven-cavity flow, the least-squares commutator preconditioner is used for both (5.2) and (5.5). In Figure 5.4, we again plot the number of GMRES steps and the shift for each iteration of Algorithm 12. The number of GMRES steps needed for solving (5.2) is about 250 for $Re = 2000$ and about 375 for $Re = 4000$, and it only varies slightly from one iteration to another. These numbers are expected from the rightmost curve of Figures 5.2(b) and 5.2(d). As observed in Figure 5.3, the number of GMRES steps needed for (5.5) oscillates with the shift and is often much smaller than that required by the solution of (5.2). The majority (approximately 75%) of the GMRES steps are again used to solve (5.2).

Next, we test the standard Krylov subspace method (Algorithm 11) with iterative linear solves on the same set of problems. Recall from section 5.1 that one solve of (5.2) is needed at each iteration of Algorithm 11. We continue to use (5.9) as the stopping criterion for the linear solve (5.2) and (5.11) as the stopping criterion for



(a) iteration count ($Re = 2000$)

(b) shifts ($Re = 2000$)

(c) iteration count ($Re = 4000$)

(d) shifts ($Re = 4000$)

Figure 5.4: Algorithm 12 with an iterative linear solver applied to driven-cavity flow ($\bullet$: the number of GMRES steps needed for solving (5.2), $\circ$: the number of GMRES steps needed for solving (5.5), $\square$: the shift in (5.5))

the Lyapunov solve. Numerical results of Algorithm 11 for flow over an obstacle at $Re = 200$ are reported in Figure 5.5. This figure shows that the number of GMRES steps needed to solve (5.2) is roughly the same (about 70) in each iteration of Algorithm 11, as observed for Algorithm 12 in Figure 5.3(a). (We can generate such a figure for the other three examples as well. They look very similar to Figure 5.5 and are therefore omitted here.) In Chapters 3 and 4, the efficiency of the two Lyapunov solvers was compared by examining how fast the residual norm $\|R\|_F$ of the Lyaupnov equation (5.1) decays as the dimension of the Krylov subspace grows (see Figure

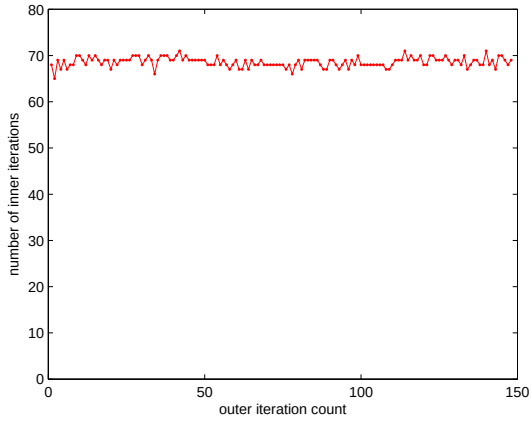
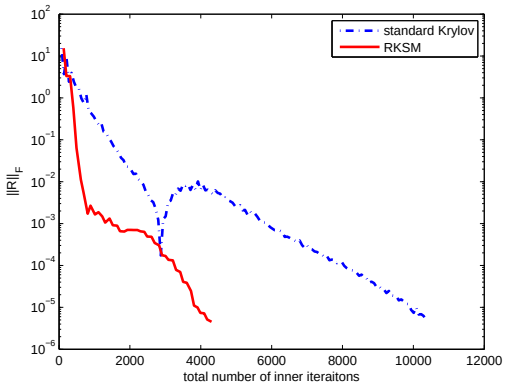
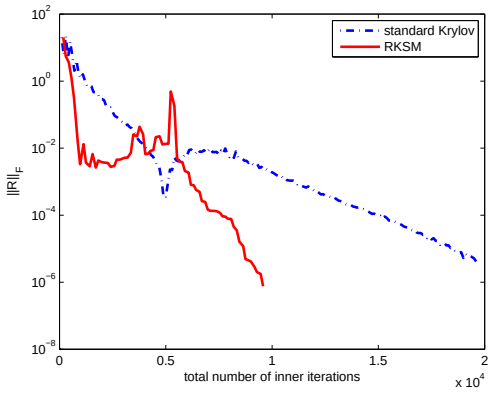


Figure 5.5: Algorithm 11 with an iterative linear solver applied to flow over an obstacle at $Re = 200$ ($\bullet$: the number of GMRES steps needed for solving (5.2))

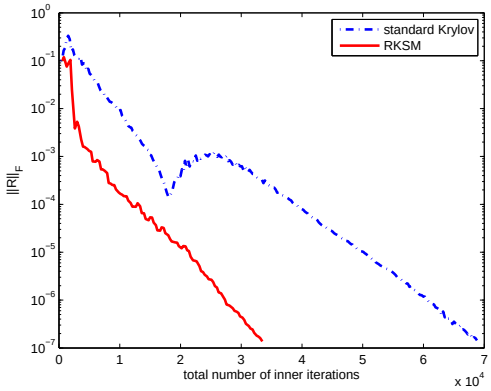
3.4 in Chapter 3 and Figures 4.3, 4.5 and 4.6 in Chapter 4). This is reasonable in the previous chapters since the cost of solving (5.5) using a direct method does not vary with the shift and therefore, the cost of generating a Krylov subspace is proportional to the dimension of the Krylov subspace for both the standard Krylov subspace method and the RKSM. Now, however, this comparison does not tell the complete story because the cost of building a Krylov subspace is no longer a simple function of the dimension of the Krylov subspace for RKSM. This point is clearly demonstrated by Figures 5.3 and 5.4. For a comparison that takes into account the fact that (5.2) and (5.5) are now solved by an iterative method, in Figure 5.6, we plot the residual norm $\|R\|_F$ associated with (5.1) against the total number of GMRES steps. As displayed in Figure 5.6, RKSM converges much faster than the standard Krylov subspace method. In all four examples we consider, to compute an approximate solution of (5.1) that satisfies (5.11), RKSM uses approximately half as many GMRES steps needed by the standard Krylov method.



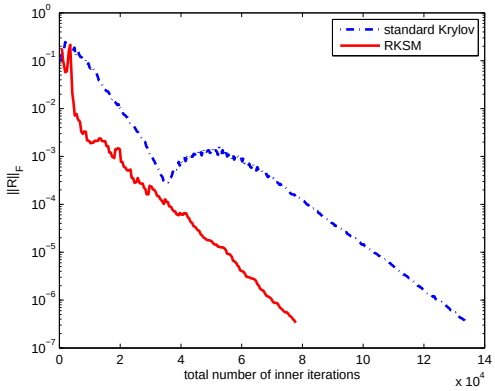
(a) obstacle ($Re = 200$)



(b) obstacle ($Re = 350$)



(c) cavity ($Re = 2000$)



(d) cavity ($Re = 4000$)

Figure 5.6: Comparison of the total number of inner iterations required by Algorithms 11 and 12

Remark. Recall from section 5.1 that compared to the standard Krylov subspace method, each iteration of RKSM requires one extra solve of (5.5). However, RKSM is still much more efficient in the examples we consider due to two reasons: first, RKSM takes much fewer iterations to converge in these examples, as shown by Figures 5.3(a) and 5.5; second, as observed earlier, solving (5.5) is on average much cheaper than solving (5.2), and therefore, an extra solve of (5.5) will not add much additional cost.

5.3 Modified Rational Krylov Subspace Method

As shown in section 5.2, the cost of the rational Krylov subspace method applied to (5.1) is dominated by that of solving the linear system (5.2). If the solution of (5.2) can somehow be avoided without harming the convergence of the Lyapunov solve, then the efficiency of this method will increase significantly. In this section, we propose a modified version of RKSM for (5.1) that achieves this goal.

Recall from section 5.1 that when RKSM (Algorithm 12) is applied to (5.1), the linear system (5.5) arises from expanding the Krylov subspace and the linear system (5.2) arises from the computation of the matrix-vector products SV_{m+1}, which in turn gives us the projection $T_m = \mathbf{V}^T S\mathbf{V}$ of S onto the Krylov subspace and the residual norm $\|R\|_F$ associated with (5.1). In addition, the eigenvalues $\left\{\widehat{\theta}_j\right\}_{j=1}^{mp}$ of T_m are needed in the generation of the next shift (see (5.3)). It is not necessary that we check the residual norm $\|R\|_F$ in every iteration of RKSM; instead, we can do it every k iterations, where $k > 1$ is an integer. However, the question still remains that if SV_{m+1} is not computed and thus T_m cannot be built, how to generate the next shift s_{m+1} of more or less the same quality as before.

Note that we actually do not need T_m itself to compute the new shift s_{m+1}; all we need is the eigenvalues of T_m. If we can get them without constructing T_m, then the generation of the new shift will no longer be an issue. We first make the observation that if the columns of $\mathbf{V}$ span an invariant subspace of S, then the eigenvalues of $T_m = \mathbf{V}^T\left(\mathbf{A}^{-1}\mathbf{M}\right)\mathbf{V}$ and $\mathcal{T}_m = \left(\mathbf{V}^T\mathbf{A}\mathbf{V}\right)^{-1}\left(\mathbf{V}^T\mathbf{M}\mathbf{V}\right)$ coincide. (In face, T_m and $\mathcal{T}_m$ are identical in this case.) The advantage of using $\mathcal{T}_m$ is that

constructing this matrix requires only matrix-vector products with $\mathbf{A}$ and $\mathbf{M}$ and no solves. The Krylov subspace built in Algorithm 12 is in general not an invariant subspace of S. However, as its dimension increases, it will tend to one and as a result, the eigenvalues of $\mathcal{T}_m$ will converge to the eigenvalues of T_m. This point is demonstrated by the following numerical experiment. Consider again (5.1) that arises from driven-cavity flow at $Re = 2000$. We compute the eigenvalues of both T_m and $\mathcal{T}_m$ as Algorithm 12 proceeds. For $m = 25, 50, 75, 100$, the spectra of both T_m and $\mathcal{T}_m$ are plotted in Figure 5.7, in which the crosses denote the eigenvalues of T_m and the circles represent the eigenvalues of $\mathcal{T}_m$. As shown in Figure 5.7, the eigenvalues of $\mathcal{T}_m$ indeed approximate those of T_m well, especially for larger m. This suggests that replacing the eigenvalues of T_m with those of $\mathcal{T}_m$ in the computation of the new shift will not affect the asymptotic convergence rate of RKSM.

The observation above leads to Algorithm 13. In the modified algorithm,

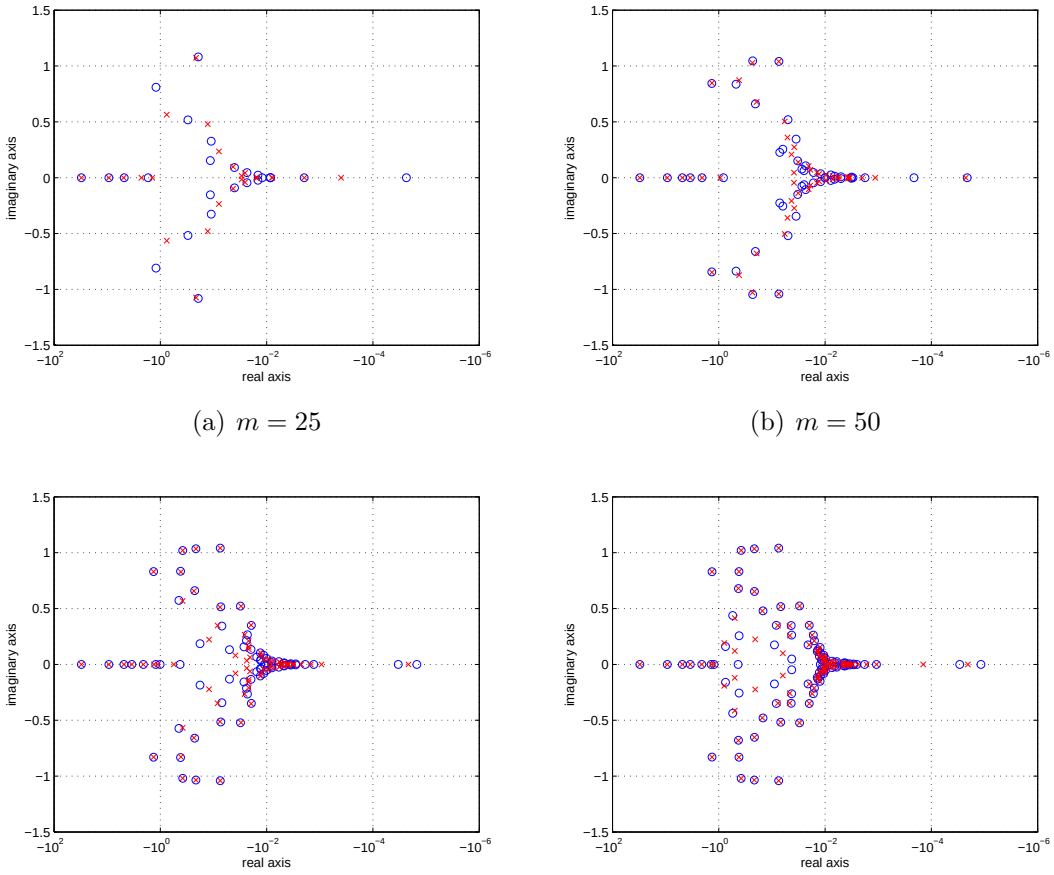
Algorithm 13: The modified rational Krylov subspace method for (5.1)

1. Given a tolerance τ, a shift s_1 and an integer $k > 1$. Let $V_1 = \mathbf{V} = P$.
2. For $m = 1, 2, \cdots$
   - 2.1. $W = (S - s_m I)^{-1} V_m$.
     for $i = 1, \ldots, m$
     $H_{i,m} \leftarrow V_i^T W$;
     $W \leftarrow W - V_i H_{i,m}$.
   - 2.2. Compute the reduced QR factorization of W: $W = V_{m+1} H_{m+1,m}$.
   - 2.3. If $mod(k, m) = 0$
     - 2.3.1. compute $T_m = \mathbf{V}^T S \mathbf{V}$ and solve the small Lyapunov equation

$$T_m Q + Q T_m^T = \left(\mathbf{V}^T P\right) C \left(\mathbf{V}^T P\right)^T;$$

     - 2.3.2. if $\|R\|_F < \tau$, then stop.
   - 2.4. Else, compute $\mathcal{T}_m = \left(\mathbf{V}^T \mathbf{A} \mathbf{V}\right)^{-1} \left(\mathbf{V}^T \mathbf{M} \mathbf{V}\right)$.
   - 2.5. $\mathbf{V} \leftarrow [\mathbf{V}, V_{m+1}]$ and compute the next shift s_{m+1}.

we compute T_m and check the convergence of the Lyapunov solve only when the



(a) $m = 25$ (b) $m = 50$

(c) $m = 75$ (d) $m = 100$

Figure 5.7: The eigenvalues of T_m (crosses) and $\mathcal{T}_m$ (circles)

iteration count m is a multiple of a prescribed number k. Consequently, (5.2) appears only in those iterations. In the iterations where T_m is not computed, we continue using (5.3) to choose the next shift; the only change that has to be made in (5.3) is to use the eigenvalues of $\mathcal{T}_m$ instead of those of T_m. The computation of $\mathcal{T}_m$ entails only two matrix-vector products $\mathbf{A}V_m$ and $\mathbf{M}V_m$ at each iteration. In fact, it only gives rise to one extra matrix-vector product $\mathbf{M}V_m$ since according to (5.4), we have to evaluate $\mathbf{A}V_m$ in step 2.1 of Algorithm 13 anyway in order to compute $(S - s_m I)^{-1} V_m$. Therefore, the cost of constructing the matrix $\mathcal{T}_m$ is negligible.

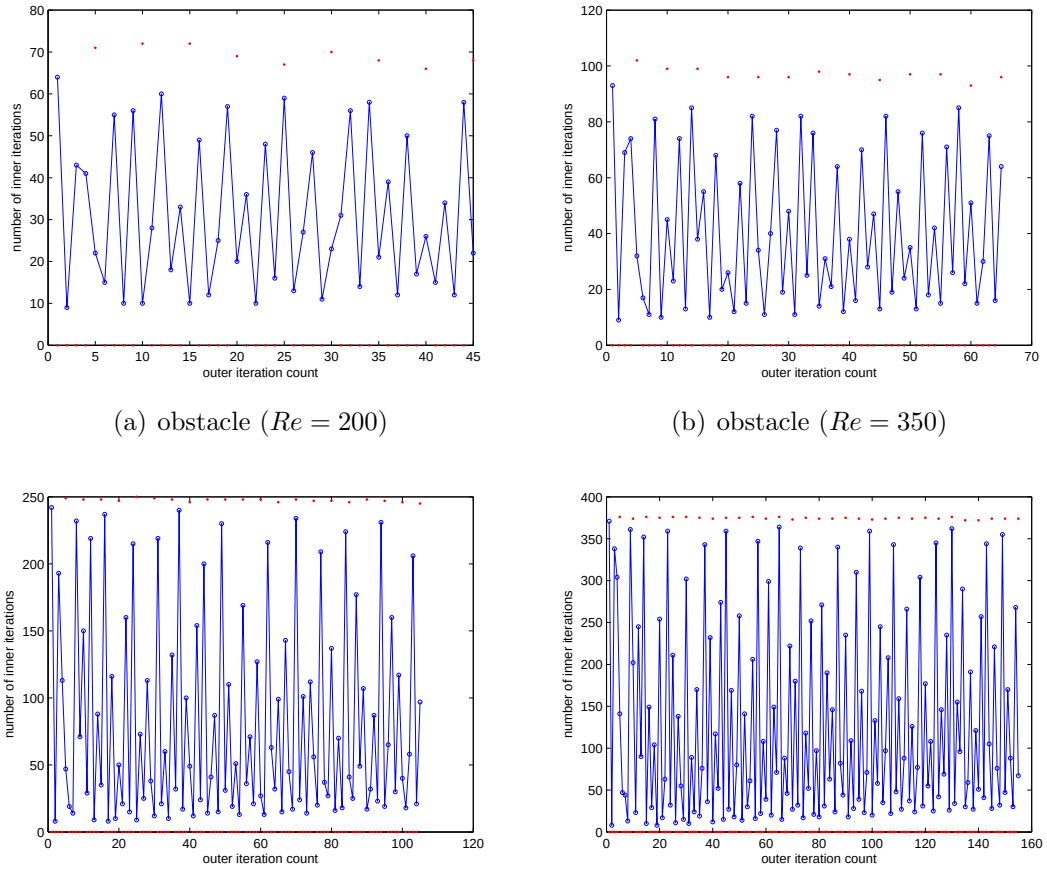
We apply Algorithm 13 (with $k = 5$) to the four examples considered in the

previous section and display the numerical results in Figure 5.8. We continue to use the stopping criteria (5.9), (5.10) and (5.11). In Figure 5.8, the dots and the circles again denote the numbers of preconditioned GMRES steps required to solve (5.2) and (5.5), respectively. Since we only compute SV_{m+1} every $k = 5$ iterations, as seen in Figure 5.8, the number of GMRES iterations taken to solve (5.2) is simply zero in many iterations. By comparing Figures 5.3, 5.4 and 5.8, we also observe that for the same example, the number of outer iterations required by Algorithm 12 and that required by Algorithm 13 are almost the same. This implies that the shifts generated in Algorithm 13 using the eigenvalues of $\mathcal{T}_m$ are essentially of the same quality as those generated in Algorithm 12 using the eigenvalues of T_m.

We also compare the total number of GMRES steps required by Algorithms 11, 12 and 13 for solving (5.1). In Figure 5.9, the residual norm of (5.1) is again plotted against the total number of GMRES steps. These residual curves show that Algorithm 13 converges much more rapidly than Algorithm 12. Compared to Algorithm 12, in order to produce an approximate solution that satisfies (5.11), Algorithm 13 takes about 50% fewer GMRES steps.

So far we have been focusing on the Lyapunov equation (5.1) arising from Lyapunov inverse iteration (Algorithm 9) applied to (4.2). As noted earlier, Lyapunov inverse iteration (Algorithm 7) applied to (3.14) gives rise to Lyapunov equations of the same type. It was observed in section 3.3.3 that in Algorithm 5.1, since we do not need to solve (5.1) very accurately, standard Krylov method is more effective than RKSM.

Suppose we want to estimate the critical Reynolds number of flow over an



(a) obstacle ($Re = 200$)

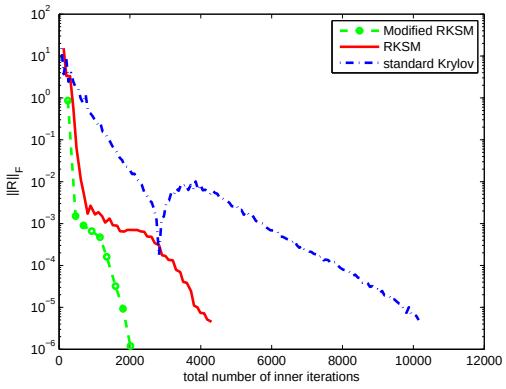
(b) obstacle ($Re = 350$)

(c) cavity ($Re = 2000$)

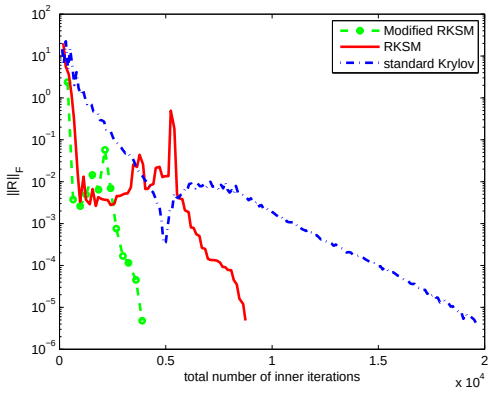
(d) cavity ($Re = 4000$)

Figure 5.8: Algorithm 13 with $k = 5$ applied to flow over an obstacle and driven-cavity flow

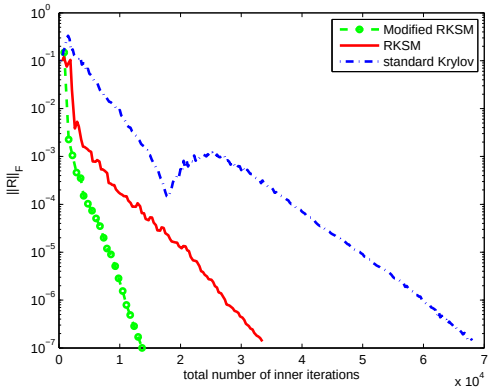
obstacle (32×128 mesh) using Algorithm 7. In addition, let $Re_0 = 320$ and $\delta = 10^{-2}$ in (3.15). (Numerical results of Algorithm 7 for this example can be found in Table A.3.) We test Algorithms 11, 12 and 13 ($k = 5$) with iterative linear solves on the Lyapunov equation arising from the second iteration of Algorithm 7. (The right-hand side of this Lyapunov equation has rank 4.) The iterative method that we use for (5.2) and (5.5) is again preconditioned GMRES. Since $\delta = 10^{-2}$ and $\|R_2^{eig}\|_F = 1.27854 \times 10^{-2}$ (see Table A.3), the stopping criterion used for the



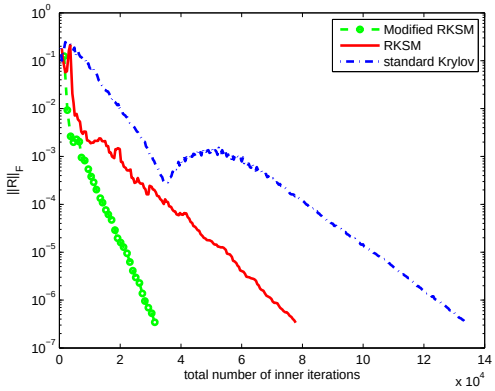
(a) obstacle ($Re = 200$)



(b) obstacle ($Re = 350$)



(c) cavity ($Re = 2000$)



(d) cavity ($Re = 4000$)

Figure 5.9: Comparison of the total number of inner iterations required by Algorithms 11, 12 and 13

Lyapunov solve is

$$\|R\|_F < \delta \|R_2^{eig}\|_F = 1.27854 \times 10^{-4}. \tag{5.12}$$

In Figure 5.10, we display the performance of the three Lyapunov solvers by plotting the residual norm $\|R\|_F$ associated with (5.1) against the total number of GMRES steps. Although RKSM is less efficient than standard Krylov method in this example, modified RKSM outperforms standard Krylov method and is again the most efficient one among the three, as observed in Figure 5.9. In order to produce an approximate solution that satisfies (5.12), modified RKSM again uses 50% fewer

GMRES steps than RKSM. As shown in section 3.3, we can get away with a δ as large as 1 in (5.12). When the tolerance is this mild, standard Krylov method and modified RKSM are almost equally efficient with standard Krylov method working slightly better.

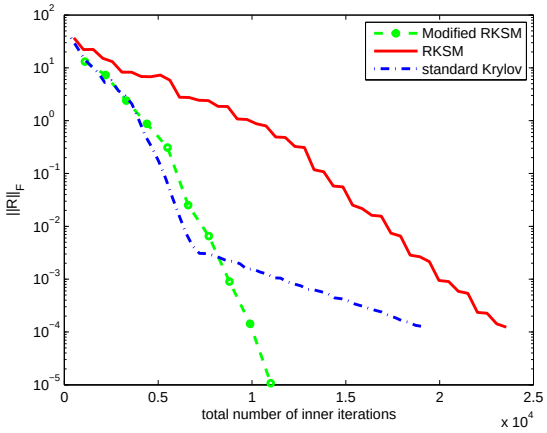


Figure 5.10: Comparison of the total number of inner iterations required by Algorithms 11, 12 and 13 (the Lyapunov equation arises from Algorithm 7 applied to (3.14))

5.4 Conclusions

In this chapter, we explore the performance of standard Krylov subspace method and rational Krylov subspace method with iterative linear solves. Different preconditioners are tested and compared on the linear systems arising from the two Lyapunov solvers. These systems can be divided into two categories: ones with structure identical to those that arise in the computation of steady states of a system of PDEs, and ones with structure like those arising from transient PDEs. We observe that the cost of solving the linear systems of the first type dominates the total cost of rational Krylov subspace method. In light of this observation, we modify

this method in such a way that solution of the first type of linear systems can mostly be avoided. The modification is simple yet effective, leading to significant savings in computational cost without degrading the convergence of the Lyapunov solver. This chapter completes the discussion on Lyapunov solvers presented in Chapters 3 and 4, in which all the linear systems arising from the Lyapunov solve were simply solved using a sparse direct method.

Chapter 6

Summary and Conclusions

In this thesis, we developed robust and efficient methods for linear stability analysis of large-scale dynamical systems. Linear stability analysis is a widely used approach for studying whether a steady state of a dynamical system is sensitive to small perturbations. It eventually boils down to determining whether the rightmost eigenvalue of the Jacobian matrix is in the left (stable) or the right (unstable) half of the complex plane, for which there lacks a robust method. A dynamical system and therefore its steady states typically depend on a physical parameter. The critical value of this parameter at which stability of the steady state is lost is of great interest to both mathematicians and engineers. The conventional approach of identifying this value is to monitor the rightmost eigenvalue of the Jacobian matrix along a parameterized path of steady states. The main contributions of this thesis are summarized as follows.

In Chapter 3, we refined a method proposed in previous work that estimates the critical parameter value using the so-called Lyapunov inverse iteration, and we applied the refined method to models of incompressible flow. This method is based on the following observation: the difference between the critical parameter value and a nearby parameter value in the stable regime is the eigenvalue with smallest modulus of a special eigenvalue problem that can be specified in the form

of a Lyapunov equation. This is a potentially easier problem since unlike for the rightmost eigenvalue, there are many reliable methods for computing eigenvalue of a matrix having smallest modulus, one of them being inverse iteration. The approach of applying inverse iteration to an eigenvalue problem with Lyapunov structure is referred to as Lyapunov inverse iteration. At each iteration of this method, a large-scale Lyapunov equation needs to be solved. We observed in our numerical experiments that solving this equation accurately can be quite expensive in the early stages of Lyapunov inverse iteration. Accordingly, we proposed an adaptive stopping criterion for the Lyapunov solve that depends on the accuracy of the eigenvalue computation. The modified Lyapunov inverse iteration equipped with this new stopping criterion leads to dramatic computational savings without affecting the convergence of the target eigenvalue. We also explored the utility of various Lyapunov solvers including the state-of-the-art rational Krylov subspace method. We found that for this particular eigenvalue problem, since the Lyapunov equations do not need to be solved very accurately, standard Krylov method is in fact more effective than the more advanced methods.

The primary limitation of the method described above is that it is only valid in the neighborhood of the critical point. In Chapter 4, we developed a robust method for computing a few rightmost eigenvalues of the Jacobian matrix using Lyapunov inverse iteration, which works at any stable point along the path of steady states. We first proved that at any point in the stable regime, the distance between the rightmost eigenvalue and the imaginary axis is the eigenvalue with smallest modulus of an eigenvalue problem with Lyapunov structure. Therefore, this distance can also

be computed using Lyapunov inverse iteration. We then presented a convergence analysis of this approach, which shows that Lyapunov inverse iteration applied to this particular eigenvalue problem will always converge in two iterations provided that the first Lyapunov equation is solved accurately enough. An efficient means of implementing Lyapunov inverse iteration was proposed, which only entails one accurate Lyapunov solve. Finally, we showed that estimates for a few rightmost eigenvalues of the Jacobian matrix can be obtained from this version of Lyapunov inverse iteration for almost no additional cost. We again compared the performance of various Lyapunov solvers for subsidiary Lyapunov systems. In contrast to the scenario in the previous chapter, what is needed in this case is one accurate Lyapunov solve, and our numerical experiments showed that rational Krylov subspace method is the method of choice.

The main cost of Lyapunov inverse iteration is solving a large Lyapunov equation iteratively at each step, which requires in turn a number of large and sparse linear solves. How efficiently these systems are solved is crucial to the performance of Lyapunov inverse iteration. The linear systems arising from Lyapunov inverse iteration can be divided into two categories: ones with structure identical to those that arise in the computation of steady states of a system of PDEs, and ones with structure like those arising from transient PDEs. In particular, when rational Krylov method is used as the Lyapunov solver, half of the linear systems are of the first category and the other half belong to the second category. We tested and compared the performance of various preconditioners developed for models of incompressible flow and observed that linear systems arising from the computation of steady states are

much more expensive to solve than those needed for time stepping. In light of this fact, we modified the rational Krylov subspace method in such a way that very few solves of the first type are needed. The modified rational Krylov subspace method achieves significant savings in computational cost. It works as well as the standard Krylov subspace method for the problem considered in Chapter 3 and is much more efficient than both the rational Krylov subspace method and the standard Krylov method for the problem studied in Chapter 4.

Appendix A

More numerical results of Algorithm 7

- Algorithm 7:

Table A.1: Algorithm 7 applied to driven-cavity flow (64×64 mesh, $Re_0 = 7700$) for $\delta = 1,\ 10^{-1},\ 10^{-2}$ in (3.15)

ℓ	$Re^{(\ell)}$	$\mu^{(\ell)}$	$\|r_\ell\|_2$	$\|R_\ell^{eig}\|_F$	$\|R_\ell^{lyap}\|_F$	d_ℓ	k_ℓ
			$\delta = 1$				
1	-33	-1.65191e-13	3.11813e-01	3.81288e+2	3.49287e+2	126	40
2	8071	2.75632i	2.28205e-04	1.51762e-1	1.51102e-1	468	170
3	7956	2.69951i	5.94238e-06	2.35053e-3	2.29411e-3	464	170
4	7941	2.69869i	1.25554e-07	7.17013e-5	6.92490e-5	440	160
5	7941	2.69871i	5.22034e-09	2.22796e-6	2.11931e-6	456	160
6	7941	2.69871i	1.58703e-10	7.03833e-8	—	—	—
					Total:	1954	
			$\delta = 10^{-1}$				
1	-33	-1.65191e-13	3.11813e-01	3.81288e+2	3.67018e+1	184	60
2	8099	3.22383i	3.63286e-04	7.42002e-2	7.21658e-3	828	260
3	8272	2.69587i	3.06514e-05	2.30574e-2	2.29802e-3	584	210
4	7940	2.69870i	9.47243e-07	3.99983e-4	3.99561e-5	628	240
5	7941	2.69871i	3.10614e-08	1.77510e-5	1.74859e-6	584	210
6	7941	2.69871i	9.92136e-10	6.43387e-7	—	—	—
					Total:	2808	
			$\delta = 10^{-2}$				
1	-33	-1.65191e-13	3.11813e-01	3.81288e+2	3.75975e+0	458	130
2	8266	2.69436i	3.23204e-05	3.91517e-2	3.74950e-4	736	240
3	7934	2.69853i	5.63398e-07	4.91929e-4	4.79841e-6	812	270
4	7941	2.69872i	3.36650e-08	2.62097e-5	2.53300e-7	804	270
5	7941	2.69871i	2.06884e-09	7.55187e-7	7.36075e-9	872	290
6	7941	2.69871i	4.18021e-11	2.58556e-8	—	—	—
					Total:	3682	

- The IRA method: $7928 \leq Re^* \leq 7929$ and $\mu \approx \pm 2.69910i$

- Algorithm 7

Table A.2: Algorithm 7 applied to driven-cavity flow (128×128 mesh, $Re_0 = 7900$) for $\delta = 1,\ 10^{-1},\ 10^{-2}$ in (3.15)

ℓ	$Re^{(\ell)}$	$\mu^{(\ell)}$	$\|r_\ell\|_2$	$\|R_\ell^{eig}\|_F$	$\|R_\ell^{lyap}\|_F$	d_ℓ	k_ℓ
$\delta = 1$							
1	-84	4.52639e-14	1.98235e-01	1.80556e+2	1.46777e+2	192	60
2	7980	2.77602i	1.49742e-05	1.11252e-2	1.08771e-2	452	160
3	8178	2.76959i	3.59421e-07	6.67463e-4	6.60235e-4	456	170
4	8170	2.76985i	1.52871e-08	1.98948e-5	1.77490e-5	456	170
5	8170	2.76986i	5.65144e-10	1.09343e-6	—	—	—
					Total:	1556	
$\delta = 10^{-1}$							
1	-84	4.52639e-14	1.98235e-01	1.80556e+2	1.80487e+1	394	120
2	8712	2.78474i	1.28825e-05	5.36405e-2	4.89405e-3	516	190
3	8195	2.76859i	3.60491e-07	1.11571e-3	1.06377e-4	540	200
4	8170	2.76988i	2.75212e-08	4.87574e-5	4.57568e-6	576	220
5	8170	2.76986i	7.86850e-10	1.66872e-6	—	—	—
					Total:	2026	
$\delta = 10^{-2}$							
1	-84	4.52639e-14	1.98235e-01	1.80556e+2	1.77080e+0	552	150
2	8279	2.78261i	6.89377e-06	2.57644e-2	2.56827e-4	732	270
3	8183	2.76926i	2.69987e-07	3.79368e-4	3.75942e-6	784	280
4	8171	2.76983i	1.97231e-08	2.36808e-5	2.30965e-7	764	270
5	8170	2.76986i	9.31122e-10	1.88047e-6	—	—	—
					Total:	2832	

- The IRA method: $8167 \leq Re^* \leq 8168$ and $\mu \approx \pm 2.76931i$

- Algorithm 7

Table A.3: Algorithm 7 applied to flow over an obstacle (32×128 mesh, $Re_0 = 320$) for $\delta = 1,\ 10^{-1},\ 10^{-2}$ in (3.15)

ℓ	$Re^{(\ell)}$	$\mu^{(\ell)}$	$\|r_\ell\|_2$	$\|R_\ell^{eig}\|_F$	$\|R_\ell^{lyap}\|_F$	d_ℓ	k_ℓ
			$\delta = 1$				
1	-51	-2.58005e-14	3.49466e-01	1.34437e+1	1.28722e+1	64	10
2	312	2.24624i	1.03971e-03	8.86864e-2	8.81434e-2	68	30
3	372	2.25768i	1.58031e-04	5.38592e-3	4.49832e-3	68	30
4	368	2.25509i	4.52672e-05	5.17687e-4	4.22587e-4	68	30
5	368	2.25445i	4.69228e-06	6.75943e-5	6.49335e-5	68	30
6	368	2.25466i	6.44924e-07	8.61572e-6	3.78203e-6	72	30
7	368	2.25466i	8.89108e-08	1.56019e-6	9.25999e-7	72	30
8	368	2.25466i	3.34159e-08	4.92662e-7	3.60198e-7	68	30
9	368	2.25466i	4.12733e-09	7.20820e-8	6.38617e-8	68	30
10	368	2.25466i	1.52598e-09	2.22553e-8	1.25189e-8	68	30
11	368	2.25466i	2.521173-10	3.57852e-9	—	—	—
					Total:	684	
			$\delta = 10^{-1}$				
1	-51	-2.58005e-14	3.49466e-01	1.34437e+1	1.097413+0	80	30
2	340	2.25959i	1.34391e-03	1.24111e-1	9.25966e-3	80	30
3	371	2.25850i	2.84080e-04	4.17974e-3	3.65068e-4	88	40
4	368	2.25419i	2.91437e-05	3.48382e-4	3.14857e-5	84	40
5	368	2.25459i	1.66116e-06	3.70041e-5	3.63305e-6	84	50
6	368	2.25470i	2.05981e-07	8.67526e-6	6.08886e-7	84	40
7	368	2.25466i	1.20058e-07	1.98096e-6	1.73313e-7	84	40
8	368	2.25466i	1.92445e-08	4.79664e-7	4.44614e-8	80	40
9	368	2.25466i	4.90524e-09	7.95222e-8	7.07355e-9	84	40
10	368	2.25466i	7.62478e-10	1.64741e-8	—	—	—
					Total:	748	
			$\delta = 10^{-2}$				
1	-51	-2.58005e-14	3.49466e-01	1.34437e+1	1.33974e-1	256	50
2	355	2.23301i	5.20369e-04	1.27854e-2	1.27569e-4	212	80
3	368	2.25539i	1.16498e-04	1.68074e-3	1.55230e-5	184	60
4	368	2.25479i	1.25632e-05	1.71052e-4	1.63911e-6	180	60
5	368	2.25465i	5.29902e-07	1.04054e-5	1.02432e-7	192	70
6	368	2.25466i	5.71101e-08	1.09042e-6	1.06047e-8	172	60
7	368	2.25466i	6.17975e-09	1.37103e-7	1.35858e-9	180	60
8	368	2.25466i	8.80752e-10	2.09028e-8	—	—	—
					Total:	1376	

- The IRA method: $366 \leq Re^* \leq 367$ and $\mu \approx \pm 2.25320i$.

- Algorithm 7

Table A.4: Algorithm 7 applied to flow over an obstacle (64×256 mesh, $Re_0 = 320$) for $\delta = 1,\ 10^{-1},\ 10^{-2}$ in (3.15)

ℓ	$Re^{(\ell)}$	$\mu^{(\ell)}$	$\|r_\ell\|_2$	$\|R_\ell^{eig}\|_F$	$\|R_\ell^{lyap}\|_F$	d_ℓ	k_ℓ
			$\delta = 1$				
1	-126	-4.56508e-14	2.34469e-01	6.82009e+0	6.54563e+0	68	10
2	309	2.25503i	1.22214e-03	2.67894e-1	1.47060e-1	60	10
3	388	2.28406i	1.79532e-04	1.44874e-2	1.38221e-2	68	30
4	371	2.26307i	3.24286e-05	1.50971e-3	7.80000e-4	68	30
5	372	2.26454i	2.80872e-06	1.26709e-4	8.29876e-5	68	20
6	372	2.26439i	1.35033e-06	5.10769e-5	3.90744e-5	64	20
7	372	2.26440i	1.15382e-07	5.09132e-6	4.22834e-6	68	30
8	372	2.26441i	5.60313e-08	2.03776e-6	1.31217e-6	64	20
9	372	2.26441i	4.75726e-09	2.25479e-7	1.17966e-7	68	20
10	372	2.26441i	1.91912e-09	6.90577e-8	3.15263e-8	64	20
11	372	2.26441i	2.16132e-10	8.88705e-9	—	—	—
					Total:	660	
			$\delta = 10^{-1}$				
1	-126	-4.56508e-14	2.34469e-01	6.82009e+0	5.94744e-1	200	30
2	371	2.24869i	5.87893e-04	4.21899e-2	4.09704e-3	92	40
3	369	2.26101i	8.09361e-05	3.07705e-3	2.78703e-4	80	40
4	372	2.26488i	8.09872e-06	3.53202e-4	2.66546e-5	80	40
5	372	2.26446i	5.68983e-07	3.12131e-5	1.81808e-6	84	40
6	372	2.26440i	5.69946e-08	3.94377e-6	3.11348e-7	84	40
7	372	2.26441i	2.64065e-08	1.19487e-6	8.67071e-8	80	40
8	372	2.26441i	4.71386e-09	1.98825e-7	1.20088e-8	80	40
9	372	2.26441i	6.18183e-10	2.82239e-9	—	—	—
					Total:	780	
			$\delta = 10^{-2}$				
1	-126	-4.56508e-14	2.34469e-01	6.82009e+0	5.65414e-2	254	50
2	355	2.24026i	1.94550e-04	1.49911e-2	1.31541e-4	184	60
3	370	2.26957i	7.74197e-05	3.01197e-3	2.89703e-5	152	60
4	372	2.26422i	1.07494e-05	4.01064e-4	3.94011e-6	156	60
5	372	2.26440i	1.55387e-06	5.52818e-5	5.09448e-7	156	50
6	372	2.26442i	4.72673e-08	2.39237e-6	2.21539e-8	168	60
7	372	2.26441i	6.31077e-09	2.60586e-7	2.24572e-9	180	60
8	372	2.26441i	9.64857e-10	4.66424e-8	—	—	—
					Total:	1250	

- The IRA method: $371 \leq Re^* \leq 372$ and $\mu \approx \pm 2.26399i$

Bibliography

[1] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. SIAM, Philadelphia, 2005.

[2] A. C. Antoulas, D. C. Sorensen, and Y. Zhou. On the decay of Hankel singular values and related issues. Technical Report 01-09, Department of Computational and Applied Mathematics, Rice University, Houston, 2001. Available from `http://www.caam.rice.edu/~sorensen/Tech_Reports.html`.

[3] F. Auteri, N. Parolini, and L. Quartapelle. Numerical investigation on the stability of singular driven cavity flow. *J. Comput. Phys.*, 183:1–25, 2002.

[4] R. H. Bartels and G. W. Stewart. Algorithm 432: solution of the matrix equation $AX + XB = C$. *Comm. of the ACM*, 15:820–826, 1972.

[5] C-H. Bruneau and M. Saad. The 2D lid-driven cavity problem revisited. *Computers & Fluids*, 35:326–348, 2006.

[6] S. Chandrasekhar. *Hydrodynamic and Hydromagnetic Stability*. Oxford University Press, Oxford, 1961.

[7] K. A. Cliffe, T. J. Garratt, and A. Spence. Calculation of eigenvalues of the discretised Navier-Stokes and related equations. In J. R. Whiteman, editor, *The Mathematics of Finite Elements and Applications VII MAFELAP*, pages 479–486. Academic Press, 1990.

[8] K. A. Cliffe, T. J. Garratt, and A. Spence. Eigenvalues of block matrices arising from problems in fluid mechanics. *SIAM J. Matrix Anal. Appl.*, 15:1310–1318, 1994.

[9] K. A. Cliffe and K. H. Winters. Convergence properties of the finite-element method for Benard convection in an infinite layer. *J. Comput. Phys*, 60:346–351, 1985.

[10] V. Druskin, L. Knizhnerman, and V. Simoncini. Analysis of the rational Krylov subspace and ADI methods for solving the Lyapunov equation. *SIAM J. Numer. Anal.*, 49:1875–1898, 2011.

[11] V. Druskin, C. Lieberman, and M. Zaslavsky. On adaptive choice of shifts in rational Krylov subspace reduction of evolutionary problems. *SIAM J. Sci. Comput.*, 32:2485–2496, 2010.

[12] V. Druskin and V. Simoncini. Adaptive rational Krylov subspaces for large-scale dynamical systems. *Systems & Control Letters*, 60:546–560, 2011.

[13] H. Elman, D. Silvester, and A. Wathen. *Finite Elements and Fast Iterative Solvers*. Oxford University Press, Oxford, 2005.

[14] H. C. Elman. Preconditioning strategies for models of incompressible flow. *J. Sci. Comput.*, 25:347–366, 2005.

[15] H. C. Elman and R. S. Tuminaro. Boundary conditions in approximate commutator preconditioners for the Navier-Stokes equations. *Electron. Trans. Numer. Anal.*, 35:257–280, 2009.

[16] A. Fortin, M. Jardak, and J. Gervais. Localization of Hopf bifurcation in fluid flow problems. *Int. J. Numer. Methods Fluids*, 24:1185–1210, 1997.

[17] T. J. Garratt. *The numerical detection of Hopf bifurcations in large systems arising in fluid mechanics*. PhD thesis, University of Bath, University of Bath, UK, 1991.

[18] J. J. Gervais, D. Lemelin, and R. Pierre. Some experiments with stability analysis of discrete incompressible flows in the lid-driven cavity. *Int. J. Numer. Methods Fluids*, 24:477–492, 1997.

[19] W. Govaerts. *Numerical Methods for Bifurcations of Dynamical Equilibria*. SIAM, Philadelphia, 2000.

[20] W. Govaerts and A. Spence. Detection of Hopf points by counting sectors in the complex plane. *Numer. Math.*, 75:43–58, 1996.

[21] N. Guglielmi and M. L. Overton. Fast algorithms for the approximation of the pseudospectral abscissa and pseudospectral radius of a matrix. *SIAM J. Matrix Anal. Appl.*, 32:1166–1192, 2011.

[22] S. J. Hammarling. Numerical solution of the stable, non-negative definite Lyapunov equation. *IMA J. Numerical. Anal.*, 2:303–323, 1982.

[23] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, 1991.

[24] I. M. Jaimoukha and E. M. Kasenally. Krylov subspace methods for solving large Lyapunov equations. *SIAM J. Numer. Anal.*, 31:227–251, 1994.

[25] J. R. Koseff and R. L. Street. The lid-driven cavity flow: A synthesis of qualitative and quantitative observations. *J. Fluids Eng.*, 106:390C398, 1984.

[26] J. R. Koseff and R. L. Street. On end wall effects in a lid driven cavity flow. *J. Fluids Eng.*, 106:385–389, 1984.

[27] J. R. Koseff and R. L. Street. Visualization studies of a shear driven three-dimensional recirculating flow. *J. Fluids Eng.*, 106:21–29, 1984.

[28] K. Meerbergen and D. Roose. Matrix transformations for computing rightmost eigenvalues of large sparse non-symmetric eigenvalue problems. *IMA J. Numer. Anal.*, 16:297–346, 1996.

[29] K. Meerbergen and A. Spence. Inverse iteration for purely imaginary eigenvalues with application to the detection of Hopf bifurcation in large scale problems. *SIAM J. Matrix Anal. Appl.*, 31:1982–1999, 2010.

[30] B. N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice Hall Series in Computational Mathematics. Prentice Hall, New Jersey, 1980.

[31] T. Penzl. Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case. *Systems Control Lett.*, 40:139–144, 2000.

[32] W. C. Rheinboldt. *Numerical Analysis of Parametrized Nonlinear Equations*, volume 7 of *The University of Arkansas Lecture Notes in the Mathematical Sciences*. J. Wiley and Sons, New York, 1986.

[33] R.Iwatsu, K. Ishii, T. Kawanura, and K. Kuwahara. Numerical simulation of three-dimensional flow structure in a driven cavity. *Fluid Dyn. Res.*, 5:173–189, 1989.

[34] M. Robbé, M. Sadkane, and A. Spence. Inexact inverse subspace iteration with preconditioning applied to non-Hermitian eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 31:92–113, 2009.

[35] V. A. Romanov. Stability of plane-parallel Couette flow. *Funct. Anal. Appl.*, 7:137–146, 1973.

[36] A. Ruhe. Rational Krylov sequence methods for eigenvalue computation. *Lin. Alg. Appl.*, 58:391–405, 1984.

[37] A. Ruhe. The rational Krylov algorithm for nonsymmetric eignevalue problems. III: complex shifts for real matrices. *BIT*, 34:165–176, 1994.

[38] Y. Saad. Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices. *Lin. Alg. Appl.*, 34:269–295, 1980.

[39] Y. Saad. Numerical solution of large Lyapunov equations. In M. A. Kaashoek, J. H. van Schuppen, and A. C. Ran, editors, *Signal Processing, Scattering, Operator Theory, and Numerical Methods*, volume III of *Proceedings of the International Symposium MTN-89*, pages 503–511, Boston, 1990. Birkhauser.

[40] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Manchester University Press, Manchester, 1992.

[41] P. N. Shankar and M. D. Deshpande. Fluid mechanics in the driven cavity. *Annu. Rev. Fluid Mech.*, 32:93–136, 2000.

[42] V. Simoncini. A new iterative method for solving large-scale Lyapunov matrix equations. *SIAM J. Sci. Comput.*, 29:1268–1288, 2007.

[43] D. C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13:357–385, 1992.

[44] G. W. Stewart. *Matrix Algorithms Volume II: Eigensystems*. SIAM, Philadelphia, 2001.

[45] G. I. Taylor. Stability of a viscous liquid contained between two rotating cylinders. *Phil. Trans. Roy. Soc. A*, 223:289–343, 1923.

[46] N . Tillmark and P . H . Alfredsson. Experiments on transition in plane Couette flow. *J. Fluid Mech.*, 235:89–102, 1992.

[47] S. Timme, K. J. Badcock, M. Wu, and A. Spence. Lyapunov inverse iteration for stability analysis using computational fluid dynamics. 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference⟨BR⟩20th AIAA/ASME/AHS Adaptive Structures Conference⟨BR⟩14th AIAA, Honolulu, Hawaii, April 2012. American Institute of Aeronautics and Astronautics.

[48] L. N. Trefethen and M. Embree. *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*. Princeton University Press, Princeton, 2005.

[49] J. S. Turner. *Buoyancy Effects in Fluids*. Cambridge University Press, Cambridge, 1973.

[50] P. A. Wedin. On angles between subspaces of a finite dimensional inner product space. In B. Kagstrom and A. Ruhe, editors, *Matrix Pencils*, volume 973 of *Lecture Notes in Mathematics*, pages 263–285. Springer-Verlag, Berlin, 1983.