University of Maryland Department of Computer Science TR-4975
University of Maryland Institute for Advanced Computer Studies TR-2011-04
March 2011

LYAPUNOV INVERSE ITERATION FOR IDENTIFYING HOPF BIFURCATIONS IN MODELS OF INCOMPRESSIBLE FLOW*

HOWARD C. ELMAN†, KARL MEERBERGEN‡, ALASTAIR SPENCE§, AND MINGHAO WU¶

Abstract. The identification of instability in large-scale dynamical systems caused by Hopf bifurcation is difficult because of the problem of identifying the rightmost pair of complex eigenvalues of large sparse generalized eigenvalue problems. A new method developed in [Meerbergen and Spence, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1982-1999] avoids this computation, instead performing an inverse iteration for a certain set of real eigenvalues and that requires the solution of a large-scale Lyapunov equation at each iteration. In this study, we refine the Lyapunov inverse iteration method to make it more robust and efficient, and we examine its performance on challenging test problems arising from fluid dynamics. Various implementation issues are discussed, including the use of inexact inner iterations and the impact of the choice of iterative solution for the Lyapunov equations, and the effect of eigenvalue distribution on performance. Numerical experiments demonstrate the robustness of the algorithm.

1. Introduction.

Consider the dynamical system

$$\mathbf{M}u_t = f(u, \alpha) \tag{1.1}$$

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 finite element discretization of partial differential equations (PDEs) where 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 u denote the steady-state solution to (1.1), *i.e.*, $u_t = 0$. We are interested in the *stability* of u: if a small perturbation $\delta(0)$ is introduced to u at time $t = 0$, does $\delta(t)$ grow with time, or does it decay? Let the *solution path* of the equilibrium equation $f(u, \alpha) = 0$ be the following set: $\mathcal{S} = \{(u, \alpha)|f(u, \alpha) = 0\}$. $\mathcal{S}$ can be computed using numerical continuation techniques (see, for example, [12]). It is often the case that as the parameter α varies, there exists a point $(u^*, \alpha^*) \in \mathcal{S}$ at which the steady-state solution u changes from being stable to unstable. An important problem in applications is to find this *critical* parameter value α^* assuming that (a portion of) $\mathcal{S}$ is known. For a fixed value of α, linear stability of the steady-state solution is determined by the spectrum of the eigenvalue problem

$$\mathbf{A}x = \mu\mathbf{M}x \tag{1.2}$$

where $\mathbf{A} = \frac{\partial f}{\partial u}(u(\alpha), \alpha)$ is the Jacobian matrix of f evaluated at α. If all the eigenvalues of (1.2) have strictly negative real part, then u is a stable steady solution; if some eigenvalues of (1.2)

*This work was supported in part by the U. S. Department of Energy under grant DEFG0204ER25619; the U. S. National Science Foundation under grant CCF0726017; the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State Science Policy Office; and the Research Council K. U. Leuven grants CoE EF/05/006 and OT/10/038.

†Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742 (elman@cs.umd.edu).

‡Department of Computer Science, Katholieke Universiteit Leuven, 3001 Heverlee (Leuven), Belgium (Karl.Meerbergen@cs.kuleuven.be).

§School of Mathematical Sciences, University of Bath, Bath, BA2 7AY, United Kingdom (a.spence@maths.bath.ac.uk).

¶Applied Mathematics & Statistics, and Scientific Computation Program, Department of Mathematics, University of Maryland, College Park, MD 20742 (mwu@math.umd.edu).

have nonnegative real part, then u is unstable. Therefore, a change of stability can be detected by monitoring the rightmost eigenvalues in the complex plane of (1.2) while marching along $\mathcal{S}$.

A steady-state solution may lose its stability in one of two ways: either the rightmost eigenvalue of (1.2) is real and passes through zero from negative to positive as α varies, as in the case of a fold point or a symmetry-breaking bifurcation point [12], or (1.2) has a complex pair of rightmost eigenvalues and they cross the imaginary axis as α varies, which leads to a Hopf bifurcation with the consequent birth of periodic solutions of (1.1). The first case is easier to detect because the rightmost eigenvalue is real and close to zero and there are many methods that are reliable for computing eigenvalues near a target, for example, the shift-and-invert Arnoldi method. However, methods like shift-and-invert Arnoldi may fail to detect the instability in the second case unless good estimates of the rightmost eigenvalues are available, which is not the case in general.

Guckenheimer *et al.* [13,14] proposed a method that computes Hopf points without computing the rightmost eigenvalues of (1.2) with $\mathbf{M} = \mathbf{I}$, the identity matrix of order n. Their method is based on the following property of the Kronecker sum $\mathbf{A} \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{A}$ (assuming $\mathbf{A}$ is nonsingular): it has a double zero eigenvalue if and only if $\mathbf{A}x = \mu x$ has one and only one pair of eigenvalues that sum to zero. The method uses the equilibrium equation $f(u, \alpha) = 0$ together with the condition $\det(\mathbf{A}(u, \alpha) \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{A}(u, \alpha)) = 0$, where $\mathbf{A}(u, \alpha)$ denotes the Jacobian matrix $\frac{\partial f}{\partial u}(u, \alpha)$, as the defining system of Hopf points and Newton's method is used to solve for the roots (u^*, α^*) of this system. Unfortunately, this algorithm requires the solution of linear systems of order n^2 where n is the order of $\mathbf{A}$ and $\mathbf{M}$; therefore it is not suitable for large-scale problems in which n is already large. Nonetheless, based on this approach, Meerbergen and Spence [18] proposed a method that estimates the critical parameter value without computing the rightmost eigenvalues of (1.2) or working with the the Kronecker sum of order n^2 directly. Estimates of the rightmost eigenvalues can be obtained as byproducts.

The aims of this paper are: (i) to further understand and refine the method discussed in [18] 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.

Throughout this paper, 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 we study are also applicable to the case where instability is caused by a real eigenvalue crossing the imaginary axis. Assume that we are currently at a stable point (u_0, α_0) on the solution path $\mathcal{S}$. Let $(u^*, \alpha^*) \in \mathcal{S}$ be an unknown point in the neighborhood of (u_0, α_0). Then the Jacobian matrix $\mathbf{A}^* = \mathbf{A}(\alpha^*)$ at the unknown point can be approximated using information available to us: $\mathbf{A}^* \approx \mathbf{A}(\alpha_0) + (\alpha^* - \alpha_0)\frac{d\mathbf{A}}{d\alpha}(\alpha_0) = \mathbf{A} + \lambda\mathbf{B}$, where $\mathbf{A}$, $\mathbf{B}$ are known, and $\lambda = \alpha - \alpha_0$ is an unknown quantity that characterizes the distance from the current point to the unknown point. For simplicity, assume $\mathbf{A}^* = \mathbf{A} + \lambda\mathbf{B}$. To detect the Hopf point at which stability is lost, *i.e.*, the Hopf point closest to (u_0, α_0), we want to compute the λ closest to zero such that

$$(\mathbf{A} + \lambda\mathbf{B})x = \mu\mathbf{M}x \tag{1.3}$$

has eigenpairs $(\mu i, x)$ and $(-\mu i, \overline{x})$. Using the equivalence between equations involving Kronecker products and linear matrix equations, it is shown in [18] that this is equivalent to finding the λ closest to zero such that

$$\mathbf{M}Z\mathbf{A}^T + \mathbf{A}Z\mathbf{M}^T + \lambda(\mathbf{M}Z\mathbf{B}^T + \mathbf{B}Z\mathbf{M}^T) = 0 \tag{1.4}$$

where $Z \in \mathbb{R}^{n\times n}$ is nonzero. Once λ is known, the critical parameter value α^* can be estimated as

$\alpha_0 + \lambda$, and a corresponding estimate of μ in (1.3) can also be found easily. These estimates could be used as starting values in an algorithm for the accurate calculation of a Hopf point (see [12]).

Consider a special case of (1.1), the Navier-Stokes equations governing viscous impressible flow,

$$\begin{aligned} u_t &= \nu\nabla^2 u - u\cdot\nabla u - \nabla p \\ 0 &= \nabla\cdot u, \end{aligned} \tag{1.5}$$

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 instead of the viscosity. Mixed finite element discretization of (1.5) gives rise to the following Jacobian matrix and mass matrix [9]

$$\mathbf{A} = \begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix},\ \mathbf{M} = \begin{bmatrix} -G & 0 \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{n\times n} \tag{1.6}$$

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 paper, we apply the method proposed in [18] to detect the Hopf point at which a steady-state solution of (1.5) loses its stability.

The plan for the rest of the paper is as follows. In section 2, we review the *Lyapunov inverse iteration* method proposed by Meerbergen and Spence in [18]. In section 3, we discuss a block Krylov method for solving large-scale Lyapunov equations with low-rank right-hand side. In section 4, we propose an inverse iteration with inexact Lyapunov solvers, which is based on the ideas in [19]. In section 5, the method proposed in section 4 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. Finally, in section 6, we make some concluding observations.

2. Review of the Lyapunov inverse iteration. In this section we review the algorithm for detecting Hopf points proposed in [18] and the mathematical theory on which the algorithm is built. The following theorem is the main theoretical motivation for the techniques in [18]:

THEOREM 2.1. *Assume both* $\mathbf{A}$ *and* $\mathbf{M}$ *are nonsingular. Then the following two statements are equivalent:*

1. $\mathbf{A}\otimes\mathbf{M} + \mathbf{M}\otimes\mathbf{A}$ *has a double zero eigenvalue corresponding to the eigenvector* $\xi_1 x_1\otimes x_2 + \xi_2 x_2\otimes x_1$ *for any* $\xi_1,\ \xi_2 \in \mathbb{C}$*;*
2. $\mathbf{A}x = \mu\mathbf{M}x$ *has one and only one pair of simple eigenvalues* μ, $-\mu$*, which sum to zero and correspond to the eigenvectors* x_1 *and* x_2*, respectively.*

The special case $\mathbf{M} = \mathbf{I}$ is considered in [13,14], and the proof for the general case can be found in [18]. We continue to assume that $\mathbf{M}$ is nonsingular. In addition, assume that there is no other pair of eigenvalues of (1.3) that sums to zero except for the pure imaginary pair at a Hopf point of (1.1). According to Theorem 2.1, if (1.3) has the eigenpairs $(\mu i, x)$ and $(-\mu i, \overline{x})$ for some λ, then for the same λ, the $n^2\times n^2$ matrix $(\mathbf{A}+\lambda\mathbf{B})\otimes\mathbf{M}+\mathbf{M}\otimes(\mathbf{A}+\lambda\mathbf{B})$ has a double zero eigenvalue associated with the eigenvector $\xi_1 x\otimes\overline{x}+\xi_2\overline{x}\otimes x$, and the converse is also true. Therefore, the problem we are interested in, *i.e.*, finding λ closest to zero such that (1.3) has a conjugate pair of pure imaginary eigenvalues, is equivalent to finding λ closest to zero such that $(\mathbf{A}+\lambda\mathbf{B})\otimes\mathbf{M}+\mathbf{M}\otimes(\mathbf{A}+\lambda\mathbf{B})$ has a double zero eigenvalue. An alternative way to state the latter problem is: find the eigenvalue λ

closest to zero for the $n^2 \times n^2$ generalized eigenvalue problem

$$(\Delta_1 + \lambda\Delta_0)z = 0 \tag{2.1}$$

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 the corresponding eigenvector of this λ is $z = \xi_1 x \otimes \overline{x} + \xi_2 \overline{x} \otimes x$. One standard approach for computing this eigenvalue is to use an iterative method such as inverse iteration for (2.1). This approach is obviously impractical for large-scale problems since inverse iteration requires solution of linear systems with coefficient matrix Δ_1, which has order n^2. We can use properties of Kronecker products to rewrite (2.1) 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 [16], p. 244). Then it is known (see [16], p. 255) that (2.1) is equivalent to (1.4). Therefore, finding λ closest to zero for (2.1) is equivalent to finding λ closest to zero for (1.4). Because of the relationship between (2.1) and (1.4), we will refer to λ an eigenvalue and Z an eigenvector of (1.4). The following theorem from [18] describes the properties of Z:

THEOREM 2.2. *Assume that λ is a real eigenvalue of (2.1). If (2.1) has eigenpairs $(\mu i, x)$ and $(-\mu i, \overline{x})$, then (1.4) has a real symmetric eigenvector of rank two, namely, $Z = xx^* + \overline{x}x^T$, which is unique up to a scalar factor and is semi-definite, and a unique skew-symmetric eigenvector of rank two, namely, $Z = xx^* - \overline{x}x^T$.*

It is suggested in [18] that we should restrict our computation to the real symmetric eigenspace of (1.4). Under this restriction, the eigenvalue of interest is simple. The corresponding eigenvector, which is symmetric and of rank two, has a natural representation in the form of a truncated eigenvalue decomposition $Z = VDV^T$, where $V \in \mathbb{R}^{n\times 2}$ is orthonormal and $D \in \mathbb{R}^{2\times 2}$ is diagonal. By Theorem 2.2, $span\{V\} = span\{x, \overline{x}\}$. Therefore, once we find the eigenvalue λ closest to zero and its eigenvector Z for (1.4), the rightmost eigenvalues of (1.3) can be found easily by solving the 2×2 problem

$$V^T(\mathbf{A} + \lambda\mathbf{B})Vy = \mu V^T\mathbf{M}Vy \tag{2.2}$$

The associated eigenvectors are $x = Vy, \overline{x} = V\overline{y}$. To find the eigenvalue closest to zero for (1.4), a version of inverse iteration can be applied:

Algorithm 1 (Inverse Iteration for (1.4))

1. Given $V_1 \in \mathbb{R}^n$ with $\|V_1\|_2 = 1$ and $D_1 = 1$, let $Z_1 = V_1 D_1 V_1^T$.
2. For $j = 1, 2, \ldots$
   2.1. Compute the eigenvalue approximation[1]

$$\lambda_j = -\frac{trace(\widetilde{A}_j^T D_j \widetilde{M}_j D_j + \widetilde{M}_j^T D_j \widetilde{A}_j D_j)}{trace(\widetilde{B}_j^T D_j \widetilde{M}_j D_j + \widetilde{M}_j^T D_j \widetilde{B}_j D_j)} \tag{2.3}$$

where

$$\widetilde{A}_j = V_j^T \mathbf{A} V_j,\ \widetilde{B}_j = V_j^T \mathbf{B} V_j,\ \widetilde{M}_j = V_j^T \mathbf{M} V_j \tag{2.4}$$

[1]The Rayleigh quotient (2.3) can be derived using a property of Kronecker products (see [16], p. 252, Exercise 25).

2.2. If (λ_j, Z_j) is accurate enough, then stop.
2.3. Else, solve

$$\mathbf{A}Y_j\mathbf{M}^T + \mathbf{M}Y_j\mathbf{A}^T = F_j \tag{2.5}$$

in factored form $Y_j = V_{j+1}D_{j+1}V_{j+1}^T$ where $F_j = \mathbf{B}Z_j\mathbf{M}^T + \mathbf{M}Z_j\mathbf{B}^T$
2.4. Normalize: $D_{j+1} \leftarrow D_{j+1}/\|D_{j+1}\|_F$. Let $Z_{j+1} = V_{j+1}D_{j+1}V_{j+1}^T$.

If $\mathbf{A}$ is nonsingular, then (2.5) is equivalent to the *Lyapunov equation*

$$SY_j + Y_jS^T = \mathbf{A}^{-1}F_j\mathbf{A}^{-T} \tag{2.6}$$

where $S = \mathbf{A}^{-1}\mathbf{M}$. Let $rank(Z_j) = k$; it is reasonable to assume that $k \ll n$ (see [18]). The right-hand side of (2.6) can be represented by its truncated eigenvalue decomposition

$$\mathbf{A}^{-1}F_j\mathbf{A}^{-T} = P_jC_jP_j^T \tag{2.7}$$

which has rank at most $2k$ and is easy to compute.[2] Since we assume that $\mathbf{M}$ is nonsingular and the point (u_0, α_0) is in the stable regime, all the eigenvalues of S lie in the left half of the complex plane. This guarantees that (2.6) has a unique solution (see [1], Chapter 6).

Theorem 2.2 implies that Z has rank 2, so when Z_j has converged, the right-hand side of (2.6), namely (2.7), has rank 4. For efficient computation of (2.6), we would like to work with $k = 2$. However, in the first few iterations, when Z_j has not converged yet, k can be much larger than 2 (although $k \ll n$). A rank-reduction procedure is introduced in [18] to guarantee a small, fixed k. Before step 2.2 in Algorithm 2, we project the eigenvalue problem (1.4) onto the subspace V_j. This leads to the $k \times k$ eigenvalue problem

$$\widetilde{M}_j\widetilde{Z}_j\widetilde{A}_j^T + \widetilde{A}_j\widetilde{Z}_j\widetilde{M}_j^T + \widetilde{\lambda}_j(\widetilde{M}_j\widetilde{Z}_j\widetilde{B}_j^T + \widetilde{B}_j\widetilde{Z}_j\widetilde{M}_j^T) = 0 \tag{2.8}$$

where $\widetilde{A}_j,\ \widetilde{B}_j,\ \widetilde{M}_j$ are computed in (2.4). For $k \ll n$, equation (2.8) can be solved using Algorithm 1 with a direct Lyapunov solver (see [3], [15]) in step 2.3. According to Theorem 2.2, $\widetilde{Z}_j$ has rank 2. Let the eigenvalue decomposition of $\widetilde{Z}_j$ be $\widetilde{V}_j\widetilde{D}_j\widetilde{V}_j^T$, where $\widetilde{V}_j \in \mathbb{R}^{k\times 2}$ and $\widetilde{D}_j \in \mathbb{R}^{2\times 2}$. We update the eigenvector $Z_j = V_jD_jV_j^T$ by $(V_j\widetilde{V}_j)\widetilde{D}_j(V_j\widetilde{V}_j)^T$. The new eigenvector has rank 2 and it forces the residual of (1.4) to be orthogonal to V_j. With the rank-reduction procedure, the right hand side of (2.6) 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 inverse iteration for (1.4) now reads:

Algorithm 2 (Inverse Iteration for (1.4) with rank reduction)
1. Given $V_1 \in \mathbb{R}^n$ with $\|V_1\|_2 = 1$ and $D_1 = 1$, let $Z_1 = V_1D_1V_1^T$ and $k = 1$.
2. For $j = 1, 2, \cdots$
   2.1. Compute (2.4), and solve for the eigenvalue $\widetilde{\lambda}_j$ of (2.8) closest to zero and its eigenvector $\widetilde{Z}_j = \widetilde{V}_j\widetilde{D}_j\widetilde{V}_j^T$, where $\widetilde{V}_j \in \mathbb{R}^{k\times r}$ and $\widetilde{D}_j \in \mathbb{R}^{r\times r}$ with $r = 1$ $(j = 1)$ or 2 $(j \geq 2)$.
   2.2. Set $Z_j = \mathcal{V}_j\widetilde{D}_j\mathcal{V}_j^T$ and $\lambda_j = \widetilde{\lambda}_j$, where $\mathcal{V}_j = V_j\widetilde{V}_j$.
   2.3. If (λ_j, Z_j) is accurate enough, then stop.
   2.4. Else, solve for Y_j from

$$SY_j + Y_jS^T = P_jC_jP_j^T \tag{2.9}$$

in factored form $Y_j = V_{j+1}D_{j+1}V_{j+1}^T$.

[2] Let $T = \mathbf{A}^{-1}\mathbf{B}$; then $\mathbf{A}^{-1}F_j\mathbf{A}^{-T} = \left(\frac{\sqrt{2}}{2}[TV_j + SV_j, TV_j - SV_j]\right)\begin{bmatrix} D_j & \\ & -D_j \end{bmatrix}\left(\frac{\sqrt{2}}{2}[TV_j + SV_j, TV_j - SV_j]\right)^T$.

3. A block Krylov Lyapunov solver. In this section, we discuss the block Krylov method for solving the Lyapunov equation

$$SY + YS^T = PCP^T \tag{3.1}$$

where $S = \mathbf{A}^{-1}\mathbf{M} \in \mathbb{R}^{n\times n}$, $P \in \mathbb{R}^{n\times s}$ orthonormal, and $C \in \mathbb{R}^{s\times s}$ diagonal, with $s \ll n$. Let K be a k-dimensional subspace of $\mathbb{R}^n$ and let V be an orthonormal basis of K. Projection methods for (3.1) seek an approximate solution of the form $\widehat{Y}(Q) = VQV^T$ with $Q \in \mathbb{R}^{k\times k}$ by imposing the so-called Galerkin condition.[3] The only Q that satisfies this condition is the solution to the projected problem (see [22])

$$(V^TSV)Q + Q(V^TSV)^T = (V^TP)C(V^TP)^T \tag{3.2}$$

In the block Krylov method (see [17], [22]), the subspace K is chosen to be

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

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 (3.1) 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

$$SV = VH_m + V_{m+1}H_{m+1,m}E_m^T \tag{3.3}$$

where $V = [V_1, \cdots, V_m] \in \mathbb{R}^{n\times ms}$ is an orthonormal basis for $\mathcal{K}_m(S,P)$, $H = \begin{bmatrix} H_m \\ H_{m+1,m}E_m^T \end{bmatrix} \in \mathbb{R}^{(m+1)s\times ms}$ is the matrix of orthogonal coefficients, and $E_m \in \mathbb{R}^{ms\times s}$ is the last s columns of the identity matrix of order ms. Note that $H_m \in \mathbb{R}^{ms\times ms}$ is block upper-Hessenberg with $s\times s$ blocks $H_{i,j}$. By the Arnoldi relationship (3.3), the projected problem (3.2) is

$$H_mQ + QH_m^T = \begin{bmatrix} C & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} = \widetilde{C} \tag{3.4}$$

which, assuming $ms \ll n$, can be solved by direct methods. An algorithmic form of the block Krylov method for solving (3.1) is given below:

Algorithm 3 (the block Krylov method for (3.1))

1. Given a tolerance τ. Let $V_1 = V = P$.
2. For $m = 1, 2, \cdots$
   - 2.1. $W = SV_m$.
     for $i = 1, \ldots, m$
     $H_{i,m} \leftarrow V_i^TW$;
     $W \leftarrow W - V_iH_{i,m}$.
   - 2.2. Solve the smaller Lyapunov equation (3.4).

[3] Let the residual function be $R(Q) = S\widehat{Y}(Q) + \widehat{Y}(Q)S^T - PCP^T$. The Galerkin condition is $\langle Z, R(Q)\rangle = tr(ZR(Q)^T) = 0$ for any Z that takes the form $V\mathcal{G}V^T$ with $\mathcal{G} \in \mathbb{R}^{k\times k}$ (see [22]).

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, $V \leftarrow [V, V_{m+1}]$.

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 s linear systems of the form

$$\mathbf{A}x = \mathbf{M}y \tag{3.5}$$

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

$$R(Q) = [V, V_{m+1}] \begin{bmatrix} H_m Q + Q H_m^T - \widetilde{C} & Q E_m H_{m+1,m}^T \\ H_{m+1,m} E_m^T Q & 0 \end{bmatrix} [V, V_{m+1}]^T \tag{3.6}$$

(see [17]). By (3.4) and (3.6), $\|R(Q)\|_F = \sqrt{2}\left\|QE_m H_{m+1,m}^T\right\|_F$ which is cheap to compute. Let $Q = U\Lambda U^T$ be the eigenvalue decomposition of Q where $\Lambda = diag(\lambda_1, \lambda_2, \cdots, \lambda_m)$ holds the eigenvalues of Q, where the moduli are in decreasing order. The computed solution $\widehat{Y}(Q)$ can usually be truncated to a (much) lower rank without affecting the residual norm:

$$\widehat{Y}(Q) = (VU)\Lambda(VU)^T = (V[U_1, U_2]) \begin{bmatrix} \Lambda_1 & \\ & \Lambda_2 \end{bmatrix} (V[U_1, U_2])^T \approx \widehat{Y}(U_1 \Lambda_1 U_1^T).$$

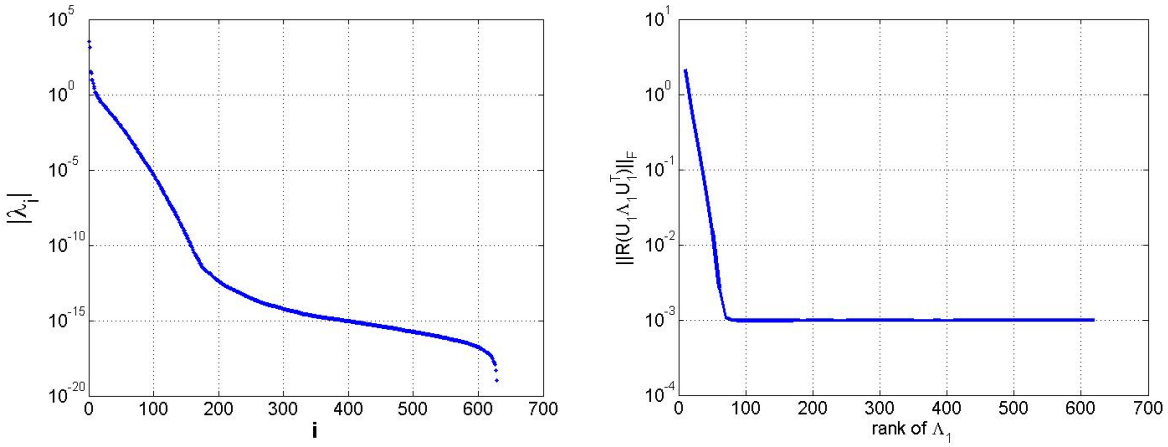
In order to do this, we increase the rank of Λ_1 until the residual norm of the truncated solution, $\|R(U_1\Lambda_1 U_1^T)\|_F$, is smaller than a prescribed tolerance τ. For example, consider the Lyapunov equation arising from the first iteration of Algorithm 2, when applied to the flow over an obstacle (details of this example are given in section 5.2). Let the tolerance $\tau = 10^{-3}$. The solution computed by Algorithm 3 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 Λ_1.

In our experiments, we have observed that when applying Algorithm 2 to problems arising from fluid mechanics, solving the Lyapunov equation (3.1) accurately can be quite expensive, especially at the early stages of the computation when the eigenvector Z_j has not converged yet. In the next section, we will show that it is in fact not necessary to solve (3.1) accurately in the first few iterations of Algorithm 2.

Different choices of the subspace K lead to variants of the standard Krylov method described here, for example, the Extended Krylov Subspace Method [23] and the Rational Krylov Subspace Method [8]. A brief discussion and some numerical results of the alternative methods will be given in section 5.3.

4. Inexact inverse iteration. In this section, we first review the main results from the previous work of Robbe *et al.* [19] on inexact inverse iteration and based on their idea, we propose an inexact inverse iteration for solving the eigenvalue problem (1.4). Suppose that a cluster ($p \ll n$) of eigenvalues of $A \in \mathbb{R}^{n\times n}$ near a shift σ is wanted. The standard approach for this problem is inverse iteration, which requires the solution of p linear systems

$$(A - \sigma I)X_i = X_{i-1} \tag{4.1}$$



(a) Decay of the eigenvalues of Q

(b) Residual norm for different ranks of truncation

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

at each step. Solving (4.1) exactly can be very challenging if n is large, which is typical when A arises from discretization of two- or three-dimensional PDEs. Therefore, the system (4.1) 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 (4.1) and the outer iteration is inverse iteration for eigenvalues. For simplicity, let $p = 1$ and $\sigma = 0$, that is, suppose we are looking for the eigenvalue closest to zero. The inexact inverse iteration in this case is as follows:

Algorithm 4 (inexact inverse iteration)

1. Given a tolerance τ, $\delta > 0$ and the starting guess z_1 with $\|z_1\|_2 = 1$.
2. For $j = 1, 2, \cdots$
   - 2.1. Compute the eigenvalue estimate: $\lambda_j = z_j^T A z_j$.
   - 2.2. Set $\mathfrak{r}_j = Az_j - \lambda_j z_j$ and test convergence.
   - 2.3. Solve $Ay_j = z_j$ for y_j inexactly such that $r_j = Ay_j - z_j$ with

$$\|r_j\|_2 < \delta \|\mathfrak{r}_j\|_2. \tag{4.2}$$

   - 2.4. Normalize: $z_{j+1} = y_j / \|y_j\|_2$

Since δ is fixed, the stopping criterion (4.2) implies the following: at the early stage of the eigenvalue computation, when $\|\mathfrak{r}_j\|_2$ is still large, the inner iteration does not need to be very accurate either; as (λ_j, z_j) converges to the true solution (*i.e.*, $\|\mathfrak{r}_j\|_2$ gets smaller), (4.1) will be solved more and more accurately. It was shown in [19] 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 (1.4) closest to zero using Algorithm 2, which requires the solution of equation (2.9) at each step. Note that $\|z\|_2 = \|Z\|_F$ if $z = vec(Z)$. Moreover, for $\mathbf{A}$ nonsingular, (1.4) is equivalent to

$$SZ + ZS^T + \lambda(SZT^T + TZS^T) = 0 \tag{4.3}$$

Therefore, in Algorithm 2, the stopping criterion

$$\|R_j\|_F < \delta \|\mathfrak{R}_j\|_F \tag{4.4}$$

is used for the inner iteration (*e.g.*, Algorithm 3) for (2.9), where $\mathfrak{R}_j = SZ_j + Z_j S^T + \lambda_j (SZ_j T^T + TZ_j S^T)$ and $R_j = SY_j + Y_j S^T - P_j C_j P_j^T$. Based on Algorithm 2 and Algorithm 4, we propose the following version of inexact inverse iteration for solving (1.4):

Algorithm 5 (Inexact Inverse Iteration for (1.4) with rank reduction)

1. Given $V_1 \in \mathbb{R}^n$ with $\|V_1\|_2 = 1$ and $D_1 = 1$, let $Z_1 = V_1 D_1 V_1^T$ and $k = 1$.
   Given $\delta > 0$.
2. For $j = 1, 2, \ldots$
   - 2.1. Compute (2.4), and solve for the eigenvalue $\widetilde{\lambda}_j$ of (2.8) closest to zero and its eigenvector $\widetilde{Z}_j = \widetilde{V}_j \widetilde{D}_j \widetilde{V}_j^T$, where $\widetilde{V}_j \in \mathbb{R}^{k \times r}$ and $\widetilde{D}_j \in \mathbb{R}^{r \times r}$ with $r = 1$ ($j = 1$) or 2 ($j \geq 2$).
   - 2.2. Set $Z_j = \mathcal{V}_j \widetilde{D}_j \mathcal{V}_j^T$ and $\lambda_j = \widetilde{\lambda}_j$, where $\mathcal{V}_j = V_j \widetilde{V}_j$.
   - 2.3. Compute $\|\mathfrak{R}_j\|_F$ and test convergence.
   - 2.4. Solve for Y_j from $SY_j + Y_j S^T = P_j C_j P_j^T$ in factored form $Y_j = V_{j+1} D_{j+1} V_{j+1}^T$ such that $\|R_j\|_F < \delta \|\mathfrak{R}_j\|_F$.
   - 2.5. Truncate the solution Y_j to rank k_j: $V_{j+1} \leftarrow V_{j+1}(:, 1 : k_j)$.

Remark. An alternative choice of the pair of residuals would be $\mathfrak{R}'_j = \mathbf{M} Z_j \mathbf{A}^T + \mathbf{A} Z_j \mathbf{M}^T + \lambda_j (\mathbf{M} Z_j \mathbf{B}^T + \mathbf{B} Z_j \mathbf{M}^T)$ and $R'_j = \mathbf{A} Y_j \mathbf{M}^T + \mathbf{M} Y_j \mathbf{A}^T - F_j$, which are the residuals of (1.4) and (2.5), respectively. We prefer the choice used in Algorithm 5 because of cost considerations. $\|R_j\|_F$ is available at almost no cost due to (3.6), and since $Z_j = \mathcal{V}_j \widetilde{D}_j \mathcal{V}_j^T$ has rank two, $\mathfrak{R}_j$ has rank four and the dominant cost of computing $\|\mathfrak{R}_j\|_F$ is the solution of four systems with coefficient matrix $\mathbf{A}$, to compute $S\mathcal{V}_j$ and $T\mathcal{V}_j$.[4] In contrast, although it is trivial to compute the Frobenius norm of the rank-four $\mathfrak{R}'_j$, it can be very expensive to evaluate $\|R'_j\|_F$: by (3.6) and the relation $R'_j = \mathbf{A} R_j \mathbf{A}^T$, computing $\|R'_j\|_F$ at the m^{th} step of Algorithm 3 requires $s(m+1)$ matrix-vector products with $\mathbf{A}$, where s is the rank of the right-hand side of (2.9). If a large number of Arnoldi steps are needed, which is indeed the case in our numerical experiments, then monitoring $\|R'_j\|_F$ instead of $\|R_j\|_F$ will be much more expensive.

5. Numerical results. In this section, we apply Algorithm 5 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 [18] and show that the algorithm we propose is robust.

In the previous sections, we always assumed that the mass matrix $\mathbf{M}$ is nonsingular. However, as given by (1.6), the mass matrix in our examples is singular. This implies that (1.2) has an *infinite eigenvalue* (*i.e.*, the eigenvalue that corresponds to the zero eigenvalue of S) of multiplicity $2n_p$ (see [5]). As shown in [5], however, replacement of $\mathbf{M}$ with the nonsingular, shifted mass matrix

$$\mathbf{M}_\sigma = \begin{bmatrix} -G & \sigma B^T \\ \sigma B & 0 \end{bmatrix} \tag{5.1}$$

maps the infinite eigenvalue of (1.2) to σ^{-1} and leaves the finite ones unchanged. With a proper choice of σ, the rightmost eigenvalue(s) of (1.2) will not be changed, which means that stability analysis will not be affected. In our computations, we use the shifted mass matrix (5.1) with

[4] In order to solve (2.9), $\mathbf{A}$ has been pre-factored or a preconditioner for it has been computed.

$\sigma = -10^{-2}$ instead of $\mathbf{M}$. The infinite eigenvalues of (1.2) are mapped to -10^2, which is well away from its rightmost eigenvalues.

All numerical results were obtained using Matlab 7.8.0 (R2009a), on a PC with a 1.60 GHz processor, and 4 GB of RAM.

5.1. 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 [9]) to compute the steady-state solution of (1.5). The left plot of Figure 5.1 shows exponentially distributed streamlines of a steady solution. Studies of the critical Reynolds number Re^* for this problem show it to be around 8000 (for example, the reported value is 7998.5 in [10], 7960 in [11], between 8017.6 and 8018.8 in [2], and between 8000 and 8050 within less than 1% error in [4], *etc.*). The rightmost eigenvalues at the critical Reynolds number are also provided in [10] ($\mu \approx \pm 2.8356i$) and [11] ($\mu \approx \pm 2.837i$). The right plot of Figure 5.1 shows the eigenvalues of $\mathbf{A}x = \mu\mathbf{M}x$ at around Re^*. 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.

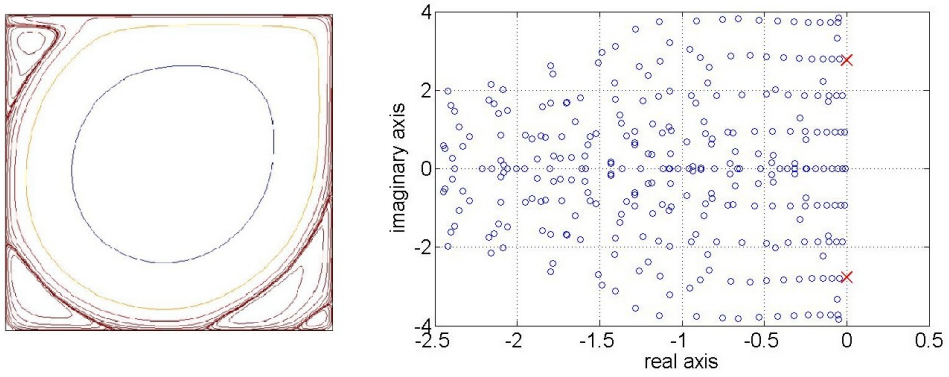


Fig. 5.1: Driven-cavity flow. Left plot: Exponentially distributed streamlines at $Re = 7500$. Right plot: The 300 eigenvalues with smallest modulus of $\mathbf{A}x = \mu\mathbf{M}x$ computed by the IRA method at $Re = 8076$ (the crosses denote the rightmost eigenvalues).

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 5 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 (4.4). Let the Reynolds number at the starting point, Re_0, be about 250 smaller than its critical value, Re^*. The goal of our tests is to find out whether Algorithm 5 is able to approximate the difference λ between the two viscosities $\nu_0 = \frac{1}{Re_0}$ and $\nu^* = \frac{1}{Re^*}$ and in turn, give us a good estimate of Re^*.[5] The computational results for the finest mesh are reported in Table 5.1. Re_j denotes the estimated value of Re^*, μ_j denotes the estimated rightmost eigenvalue of (1.3), $r_j = (A + \lambda_j B)x_j - \mu_j M x_j$ is the residual of (1.3), and R_j, $\mathfrak{R}_j$ are defined in the previous section. In addition, m_j is the rank of the solution of (2.9) before truncation, and k_j is the rank after

[5]Let $\lambda = \nu^* - \nu_0$; then once λ is approximated by Algorithm 5, Re^* can be estimated by $\frac{1}{\nu_0 + \lambda}$.

Table 5.1: Driven-cavity flow (256×256 mesh, $Re_0 = 7800$) for $\delta = 1,\ 10^{-1},\ 10^{-2}$ in (4.4)

j	Re_j	μ_j	$\|r_j\|_2$	$\|\mathfrak{R}_j\|_F$	$\|R_j\|_F$	m_j	k_j
				$\delta = 1$			
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	

truncation. The main cost of each iteration is the m_j solves of linear systems with coefficient matrix $\mathbf{A}$. The computation terminates when $\|r_j\|_2 < 10^{-9}$ is satisfied. In the first iteration, when a real, symmetric and rank-one matrix vv^T (v is a random vector in $\mathbb{R}^n$) is used as the eigenvector estimate of (1.4), the λ we computed is quite far away from its true value, causing the estimated critical Reynolds number to be nonphysical (-219). However, starting from the second iteration, λ 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 ($\|R_j\|_F < \|\mathfrak{R}_j\|_F$). Computational results for the two coarser meshes can be found in Appendix A and the same trend can be observed there.

As observed above, a commonly used method to locate the first Hopf point is to compute the rightmost eigenvalues of $\mathbf{A}x = \mu\mathbf{M}x$ 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 5. The details are as follows: for each point in the set, we compute the 250 eigenvalues with smallest modulus for $\mathbf{A}x = \mu\mathbf{M}x$ using Matlab function 'eigs' (with other parameters set to default values), which implements the *implicitly restarted Arnoldi* (IRA) method [24]. For the finest mesh, the critical Reynolds number found by this method is between 8075 and 8076, and the rightmost eigenvalues are $\mu \approx \pm 2.80905i$. This shows that Algorithm 5 yields good estimates of Re^* and μ. 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 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 $\mathbf{A}$, and typically many more. In our experience, locating Re^* by monitoring the rightmost eigenvalues along $\mathcal{S}$ is much more expensive than Algorithm 5 with $\delta = 1$.

5.2. 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 [9] for details). Again we use IFISS to compute the steady-state solution. Uniformly distributed streamlines of the steady solution are plotted in Figure 5.2a. As in the previous example, we use Q_2-Q_1 mixed finite element discretization and apply Algorithm 5 (with $\delta = 1, 10^{-1}, 10^{-2}$) on three meshes: 32×128 ($n = 9512$), 64×256 ($n = 37168$) and 128×512 ($n = 146912$).

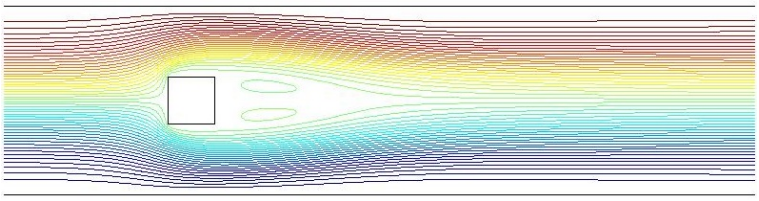
We choose Re_0 to be 50 smaller than the critical value Re^*. The computational results for the finest mesh are reported in Table 5.2. Results given by the IRA method (see Example 1) are the following: $372 \leq Re^* \leq 373$ and $\mu \approx \pm 2.26578i$. The 300 eigenvalues with smallest modulus at a Reynolds number close to Re^* are plotted in Figure 5.2b. As in the previous example, our algorithm gives good estimates of Re^* and μ. 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 Appendix B.

5.3. Discussion of Lyapunov solvers. As observed above, the efficiency of Algorithm 5 depends largely on the cost of solving the large-scale Lyapunov equation (3.1) at each iteration. In section 3, we discussed the Krylov method which searches for an approximate solution VQV^T, where V is an orthonormal basis of the Krylov subspace $\mathcal{K}_m(S, P) = span\left\{P, SP, S^2P, \cdots, S^{m-1}P\right\}$ and Q solves the small projected problem obtained by imposing the Galerkin condition. As shown in Example 5.1 (driven-cavity flow), a large Krylov subspace is needed for this method to compute an accurate enough solution even for the mild tolerance $\|R_j\|_F < \|\mathfrak{R}_j\|_F$. This deficiency leads us to the exploration of alternative Lyapunov solvers.

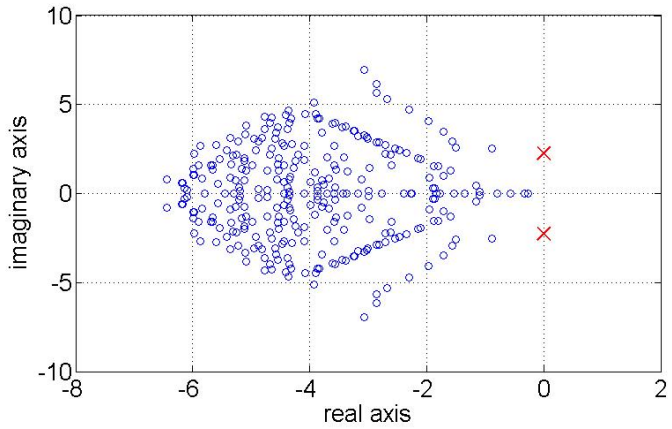
A recently developed projection method is the Rational Krylov Subspace Method (RKSM) [8]. Like the standard Krylov method, it projects a large Lyapunov equation onto a much smaller subspace, solves the small Lyapunov equation obtained by imposing the Galerkin condition and projects the solution back to the original space. In this method, the Krylov subspace is defined to be

$$\mathcal{K}_m(S, P, \mathbf{s}) = span\left\{(S - s_1 I)^{-1}P, (S - s_2 I)^{-1}(S - s_1 I)^{-1}P, \cdots, \prod_{j=0}^{m-1}(S - s_{m-j}I)^{-1}P\right\}$$

where $\mathbf{s} = [s_1, s_2, \cdots, s_m]^T \in \mathbb{C}^m$ is a vector of shifts that can be selected a priori or generated adaptively during computation. An algorithm that computes a decomposition similar to (3.3) for



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



(b) The 300 eigenvalues with smallest modulus of $\mathbf{A}x = \mu\mathbf{M}x$ computed by the IRA method at $Re = 373$ (the crosses denote the rightmost eigenvalues)

Fig. 5.2: Flow over an obstacle

$\mathcal{K}_m(S, P, \mathbf{s})$ can be found in [21]. The use of such a subspace is first introduced by Ruhe for eigenvalue computation [20], where the shifts are placed around the target eigenvalues. In [7], RKSM is used to approximate $u(t) = \exp(St)\varphi \in \mathbb{R}^n$ where $S \in \mathbb{R}^{n\times n}$ is symmetric negative definite. An adaptive approach of choosing the shifts is proposed in [7] 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 [7] that they can be restricted to the interval $[-\lambda_{max}, -\lambda_{min}]$ on the real line, where λ_{max} and λ_{min} are the largest and smallest eigenvalues of S, respectively. We present their formula for computing the next shift s_{m+1} $(m \geq 2)$ without going into detail:

$$s_m = \arg\left(\max_{s\in\mathcal{I}} \tfrac{1}{|r_m(s)|}\right),\ r_m(z) = \prod_{j=1}^{m} \frac{\left(z - \lambda_j^{(m)}\right)}{(z - s_j)} \tag{5.2}$$

where $\{\lambda_j^{(m)}\}_{j=1}^m$ are the Ritz values of S on the Krylov subspace $\mathcal{K}_m = (S, \varphi, \mathbf{s})$, $\{s_j\}_{j=1}^m$ are the shifts of previous iterations, and $\mathcal{I} = [-\lambda_{max}, -\lambda_{min}]$. In each Arnoldi step, a new pole will be added to the denominator of $r_m(s)$ and the numerator of $r_m(s)$ will be completely changed. To start the computation, the first two shifts s_1 and s_2 are set to be estimates of $-\lambda_{max}$ and

Table 5.2: Flow over an obstacle (128×512 mesh, $Re_0 = 320$) for $\delta = 1,\ 10^{-1},\ 10^{-2}$ in (4.4)

j	Re_j	μ_j	$\|r_j\|_2$	$\|\mathfrak{R}_j\|_F$	$\|R_j\|_F$	m_j	k_j
				$\delta = 1$			
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	

$-\lambda_{min}$, which must be provided by some means. In [8], it is shown that this adaptive computation of the shifts can be used to generate an efficient Krylov subspace for solving the Lyapunov equation (3.1).[6] 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 (3.1). To generate the adaptive approach to a nonsymmetric S, [8] suggests replacing $\mathcal{I} = [-\lambda_{max}, -\lambda_{min}]$ by $\mathcal{I} = [-Re_{max}(\lambda), -Re_{min}(\lambda)]$. As before, $Re_{max}(\lambda)$

[6] In solving Lyapunov equations, $\mathcal{K}_m(S, P, \mathbf{s}) = span\left\{P, (S - s_1 I)^{-1}P, \cdots, \prod_{j=1}^{m-1}(S - s_{m-j}I)^{-1}P\right\}$.

and $Re_{min}(\lambda)$ must be estimated beforehand (see [8] for a discussion). A convergence analysis of RKSM is given in [6].

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. Both $\mathbf{A}$ and $\mathbf{M}$ are large and sparse and $\mathbf{A}$ is nonsymmetric. To build a k-dimensional Krylov subspace, the standard Krylov method requires k solves with the coefficient matrix $\mathbf{A}$, and RKSM requires k solves with the coefficient matrices $\mathbf{M} - s_j\mathbf{A}$ where the s_j's do not coincide in general. When the standard Krylov method is applied, we can either pre-factor $\mathbf{A}$ or pre-compute a preconditioner for $\mathbf{A}$, whereas in RKSM, we cannot do the same thing for the coefficient matrices because they are different in each iteration. Therefore, when Krylov subspaces of the same size are generated, RKSM will be more expensive than the standard Krylov method. In addition, the Frobenius norm of the residual at each step of the standard Krylov method can be obtained by (3.6) almost for free, whereas computing this quantity at each step of RKSM requires s solves of linear systems with coefficient matrix $\mathbf{A}$ (see [8], Proposition 4.1), where s is the rank of the right-hand side of (3.1). Therefore, RKSM is only competitive when it can generate the solution with a smaller subspace.

The examples we consider here are the Lyapunov equations

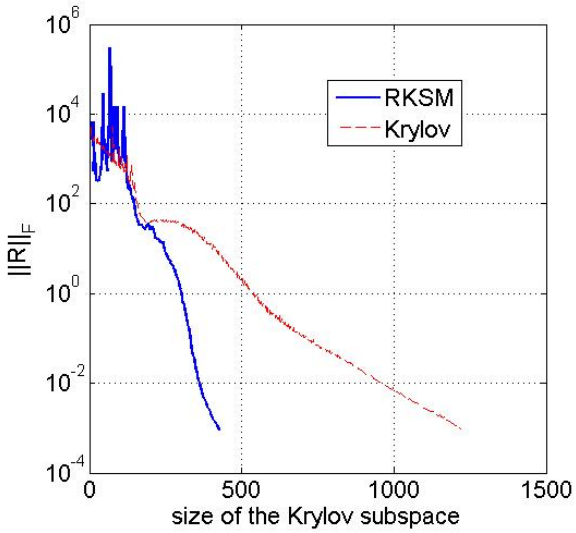
$$SY_j + Y_jS^T = P_jC_jP_j^T,\ j = 1, 2, 3 \tag{5.3}$$

arising from the first three iterations of Algorithm 5 (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 $j = 1$ and 4 for $j = 2, 3$. We compare the performance of the standard Krylov method and RKSM for solving (5.3). The coefficient matrix $\mathbf{A}$ is pre-factored in the standard Krylov method. Let the residual be R_j and the stopping criterion be $\|R_j\|_F < 10^{-3}$ for all three equations. When $j = 1$, the residual norm decay of the two methods is plotted in Figure 5.3. The ranks of the computed solutions of the two methods are reported in Table 5.3. In both examples, 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 7 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.

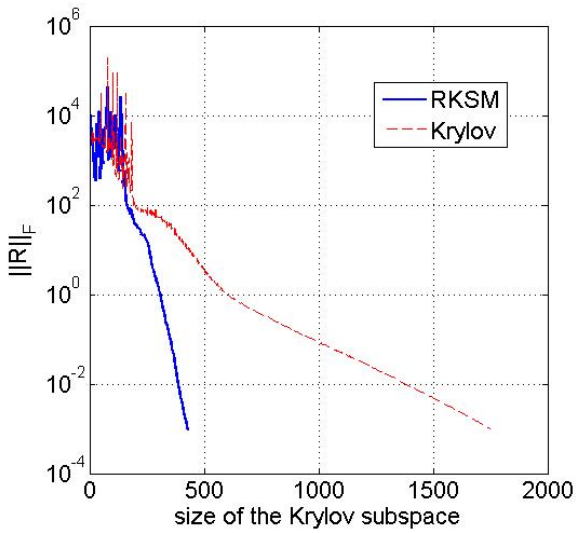
Table 5.3: Rank of the approximate solution ($j = 1$)

mesh	Krylov	RKSM
64×64	1218	426
128×128	1748	428

If the goal is to solve (3.1) accurately, RKSM is definitely the method of choice. In fact, because of its fast asymptotic convergence rate, the more accurately we want to solve the Lyapunov equation, the cheaper RKSM becomes compared with the Krylov method. However, as pointed out in section 4, solving the Lyapunov equations accurately is not of primary interest, since we only need to solve it accurately enough for the outer iteration, *i.e.*, $\|R_j\|_F < \|\mathfrak{R}_j\|_F$. In our experiments,



(a) 64×64 grid ($n = 9539$)



(b) 128×128 grid ($n = 37507$)

Fig. 5.3: Residual norm decay of the two Lyapunov solvers ($j = 1$)

$\|\mathfrak{R}_1\|_F \approx 10^2$ (see Appendix A), so we only require $\|R_1\|_F \lesssim 10^2$. From Figure 5.3, we can see that if the stopping criterion is this mild, the two methods require Krylov subspaces of almost the same size and therefore, RKSM will be the less effective choice between the two.

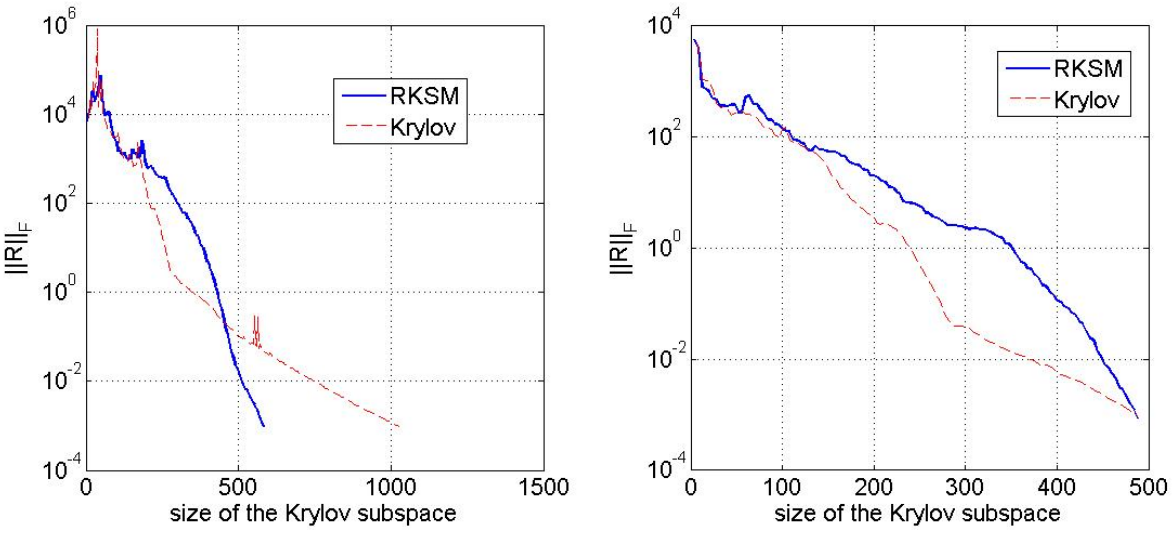
As the outer iteration proceeds, the Lyapunov equation becomes progressively easier to solve and this trend becomes more pronounced. Figure 5.4 shows the residual norm decay of the two solvers when $j = 2, 3$. RKSM has the fastest asymptotic convergence rate in both cases, and if we make the tolerance (10^{-3}) even smaller, this method will outperform the Krylov method eventually. However, as seen in the case $j = 1$, it is again more expensive than the Krylov method in the regime we are interested in ($\|R_j\|_F < \|\mathfrak{R}_j\|_F$). Note that the Krylov method gets significantly more efficient as the outer iteration proceeds.

Table 5.4: Rank of the approximate solution and CPU time, 64×64 mesh ($j = 1, 2, 3$)

j	Krylov	RKSM
1	1218 (63 min.)	426 (7 min.)
2	1024 (15 min.)	584 (8 min.)
3	488 (1 min.)	488 (5 min.)

Remark. Another projection method we explored is the Extended Krylov Subspace Method (EKSM) (see [23]). This method builds the Krylov subspace

$$\mathbf{K}_m(S, P) = \mathcal{K}_m(S, P) + \mathcal{K}_m(S^{-1}, S^{-1}P)$$



(a) $j = 2$ (b) $j = 3$

Fig. 5.4: Residual norm decay of the two Lyapunov solvers, 64×64 mesh ($j = 2, 3$)

with the hope that through the use of inverse powers of S, the Krylov subspace will carry richer spectral information. However, for all the benchmark problems above, it performed poorly in terms of both rank of the approximate solution and CPU time.

6. Conclusions. We have refined the Lyapunov inverse iteration proposed in [18] 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 as effective as the Rational Krylov Subspace Method for solving the large-scale Lyapunov systems that arise.

References.

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

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

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

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

[5] 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** (1994), pp. 1310–1318.

[6] V. Druskin, L. Knizhnerman, and V. Simoncini, *Analysis of the rational Krylov subspace and ADI methods for solving the Lyapunov equation*, Technical report, Dipartimento di Matematica, Universita di Bologna, 2010. Available from `http://www.dm.unibo.it/simoncin/list_bysub.html`.

[7] 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** (2010), pp. 2485–2496.

[8] V. Druskin and V. Simoncini, *Adaptive rational Krylov subspaces for large-scale dynamical systems*, Technical report, Dipartimento di Matematica, Universita di Bologna, 2010. Available from `http://www.dm.unibo.it/simoncin/list_bysub.html`.

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

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

[11] 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** (1997), pp. 477–492.

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

[13] J. Guckenheimer and M. Myers, *Computing Hopf bifurcations II: Three examples from neurophysiology*, SIAM J. Sci. Comput., **17** (1996), pp. 1275–1301.

[14] J. Guckenheimer, M. Myers, and B. Sturmfels, *Computing Hopf bifurcations I*, SIAM J. Numer. Anal., **34** (1997), pp. 1–21.

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

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

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

[18] 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** (2010), pp. 1982–1999.

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

[20] A. Ruhe, *Rational Krylov sequence methods for eigenvalue computation*, Lin. Alg. Appl., **58** (1984), pp. 391–405.

[21] ———, *The rational Krylov algorithm for nonsymmetric eignevalue problems. III: complex shifts for real matrices*, BIT, **34** (1994), pp. 165–176.

[22] Y. Saad, *Numerical solution of large Lyapunov equations*, Technical Report lyapunov89, RIACS, NASA Ames Research center, Moffett Field, CA, 1989. Available from `http://www-users.cs.umn.edu/~saad/reports.html`.

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

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

Appendix A. Numerical results for Example 1.

A.1. The 64×64 mesh ($n = 9539$).

- Algorithm 5

Table A.1: Driven-cavity flow (64×64 mesh, $Re_0 = 7700$) for $\delta = 1,\ 10^{-1},\ 10^{-2}$ in (4.4)

j	Re_j	μ_j	$\|r_j\|_2$	$\|\mathfrak{R}_j\|_F$	$\|R_j\|_F$	m_j	k_j
				$\delta = 1$			
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 $\mu \approx \pm 2.69910i$

A.2. The 128×128 mesh ($n = 37507$).

- Algorithm 5:

Table A.2: Driven-cavity flow (128×128 mesh, $Re_0 = 7900$) for $\delta = 1,\ 10^{-1},\ 10^{-2}$ in (4.4)

j	Re_j	μ_j	$\|r_j\|_2$	$\|\mathfrak{R}_j\|_F$	$\|R_j\|_F$	m_j	k_j
				$\delta = 1$			
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 $\mu \approx \pm 2.76931i$

Appendix B. Numerical results for Example 2.

B.1. The 32×128 mesh ($n = 9512$).

- Algorithm 5:

Table B.1: Flow over an obstacle (32×128 mesh, $Re_0 = 320$) for $\delta = 1,\ 10^{-1},\ 10^{-2}$ in (4.4)

j	Re_j	μ_j	$\|r_j\|_2$	$\|\mathfrak{R}_j\|_F$	$\|R_j\|_F$	m_j	k_j
				$\delta = 1$			
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 $\mu \approx \pm 2.25320i$.

B.2. The 64×256 mesh ($n = 37168$).

- Algorithm 5:

Table B.2: Flow over an obstacle (64×256 mesh, $Re_0 = 320$) for $\delta = 1,\ 10^{-1},\ 10^{-2}$ in (4.4)

j	Re_j	μ_j	$\|r_j\|_2$	$\|\mathfrak{R}_j\|_F$	$\|R_j\|_F$	m_j	k_j
				$\delta = 1$			
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 $\mu \approx \pm 2.26399i$