IMA Journal of Numerical Analysis (2015) Page 1 of 21
doi:10.1093/imanum/drv003

Efficient iterative algorithms for linear stability analysis of incompressible flows

Howard C. Elman
Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA
elman@cs.umd.edu

and

Minghao W. Rostami*
Department of Mathematical Sciences, Worcester Polytechnic Institute, Worcester, MA 01609, USA
*Corresponding author: mwu@wpi.edu

[Received on 6 November 2013; revised on 11 August 2014]

Linear stability analysis of a dynamical system entails finding the rightmost eigenvalue for a series of eigenvalue problems. For large-scale systems, it is known that conventional iterative eigenvalue solvers are not reliable for computing this eigenvalue. A more robust method recently developed in Elman & Wu (2013, Lyapunov inverse iteration for computing a few rightmost eigenvalues of large generalized eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, **34**, 1685–1707) and Meerbergen & Spence (2010, 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), *Lyapunov inverse iteration*, involves solving large-scale Lyapunov equations, which in turn requires the solution of large, sparse linear systems analogous to those arising from solving the underlying partial differential equations (PDEs). This study explores the efficient implementation of Lyapunov inverse iteration when it is used for linear stability analysis of incompressible flows. Efficiencies are obtained from effective solution strategies for the Lyapunov equations and for the underlying PDEs. Solution strategies based on effective preconditioning methods and on recycling Krylov subspace methods are tested and compared, and a modified version of a Lyapunov solver is proposed that achieves significant savings in computational cost.

Keywords: linear stability analysis; Lyapunov equation; rational Krylov subspace method; recycling Krylov subspaces.

1. Introduction

In this paper, we will discuss efficient computational algorithms for linear stability analysis of a large-scale dynamical system of the form

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

where $\mathbf{M} \in \mathbb{R}^{n\times n}$ is called the mass matrix and is large and sparse, $u \in \mathbb{R}^n$ is the state variable (velocity, pressure, temperature, etc.) and α is a physical parameter. Such a dynamical system arises from spatial discretization of two- or three-dimensional partial differential equations (PDEs).

Linear stability analysis is a standard approach to studying the sensitivity of a steady state $\bar{u}$ of (1.1) to small perturbations: roughly speaking, if all small perturbations introduced to $\bar{u}$ will eventually die out, then $\bar{u}$ is stable; and if some of them will grow with time, then $\bar{u}$ is considered unstable. We are especially interested in identifying the critical point $(\bar{u}_c, \alpha_c)$ at which $\bar{u}$ changes from being stable to

unstable. The linear stability of $\bar{u}$ is determined by the rightmost eigenvalue (i.e., the eigenvalue with an algebraically largest real part) of a generalized eigenvalue problem

$$J(\bar{u},\alpha)x = \mu \mathbf{M}x, \tag{1.2}$$

where $J(\bar{u},\alpha) = (\partial f/\partial u)(\bar{u},\alpha)$ is the Jacobian matrix, which is often large, sparse and, in general, non-symmetric. If the rightmost eigenvalue of (1.2) has negative real part, then $\bar{u}$ is stable; otherwise, it is unstable. At the critical point $(\bar{u}_c,\alpha_c)$, the rightmost eigenvalue of (1.2) has real part zero.

Consequently, $(\bar{u}_c,\alpha_c)$ can be located by monitoring the rightmost eigenvalue of (1.2) along a path of stable steady states. Commonly used iterative eigenvalue solvers such as Arnoldi's method and its variants (see Stewart, 2001) work well when a small set of eigenvalues of (1.2) near a given point $\sigma \in \mathbb{C}$ (called the 'shift') are sought. Thus, a good estimate for the rightmost eigenvalue of (1.2) would be an ideal choice for σ. Unfortunately, such an estimate is usually not available. In practice, it is common to choose $\sigma = 0$, that is, to compute a number of eigenvalues of (1.2) closest to zero, hoping that the rightmost one is among them. One major disadvantage of this strategy lies in its lack of robustness: a rightmost eigenvalue with large imaginary part may not be found.

Recently, a more robust method for computing the rightmost eigenvalue of (1.2) has been developed in Elman & Wu (2013). This method finds the rightmost eigenvalue of (1.2) by introducing a new eigenvalue problem in the form of a *Lyapunov equation* and computing its eigenvalue with smallest modulus. A brief description of this method is as follows. Let $\mathbf{A} = J(\bar{u}_0,\alpha_0)$, where $(\bar{u}_0,\alpha_0)$ is any point in the stable regime. In addition, let μ_1 denote the rightmost eigenvalue of $\mathbf{A}x = \mu\mathbf{M}x$ and x_1 (with $\|x\|_2 = 1$) the eigenvector associated with it. It was shown in Elman & Wu (2013) and Meerbergen & Vandebril (2012) that if $\mathbf{M}$ is nonsingular, then the eigenvalue with smallest modulus of

$$SZ + ZS^{\mathrm{T}} + \lambda(2SZS^{\mathrm{T}}) = 0, \tag{1.3}$$

where $S = \mathbf{A}^{-1}\mathbf{M}$ is $-\frac{1}{2}(\mu_1 + \bar{\mu}_1)$. Moreover, under the generic assumptions that μ_1 is simple and that, except for $\bar{\mu}_1$, no other eigenvalue of $\mathbf{A}x = \mu\mathbf{M}x$ has the same real part as μ_1, there is a unique (up to a scalar multiplier), real, and symmetric 'eigenvector' of (1.3) associated with $-\frac{1}{2}(\mu_1 + \bar{\mu}_1)$ given by $x_1x_1^* + \bar{x}_1x_1^{\mathrm{T}}$. This eigenvector is of rank 1 if μ_1 is real or rank 2 otherwise. As a result, it can be represented efficiently using its truncated eigenvalue decomposition $\mathcal{V}\mathcal{D}\mathcal{V}^{\mathrm{T}}$, where the matrix $\mathcal{V}$ consists of one or two orthonormal columns that form a basis for $\mathrm{span}\{x_1,\bar{x}_1\}$.

The observation above suggests the following approach to finding the rightmost eigenvalue of $\mathbf{A}x = \mu\mathbf{M}x$: first solve (1.3) for its eigenvalue with smallest modulus and the associated eigenvector, and then solve the small eigenvalue problem $(\mathcal{V}^{\mathrm{T}}\mathbf{A}\mathcal{V})y = \mu(\mathcal{V}^{\mathrm{T}}\mathbf{M}\mathcal{V})y$. The advantage of this strategy is that, unlike for the rightmost eigenvalue, there are many robust methods for computing an eigenvalue with smallest modulus, in particular, inverse iteration (also known as inverse power method; see Stewart (2001)). In Elman & Wu (2013), a variant of inverse iteration referred to as *Lyapunov inverse iteration* was used. This algorithm was first proposed in Meerbergen & Spence (2010) for eigenvalue problems similar in structure to (1.3); the methodology was developed to take advantage of the special low-rank structure of the 'eigenvector' Z of (1.3).

Applying Lyapunov inverse iteration to (1.3) requires solving a large-scale Lyapunov equation

$$SY + YS^{\mathrm{T}} = PCP^{\mathrm{T}} \tag{1.4}$$

at each step, where $P \in \mathbb{R}^{n\times r}$ with $r = 1$ or 2 (see Elman & Wu, 2013) consists of orthonormal column(s). Hence, the implementation of Lyapunov inverse iteration depends on solving (1.4)

efficiently. The solution to (1.4) often has low-rank approximation (see Penzl, 2000; Antoulas *et al.*, 2001; Grasedyck, 2004; Kressner & Tobler, 2010), and iterative methods for computing such an approximation (see, for instance, Saad, 1990; Simoncini, 2007; Druskin & Simoncini, 2011) entail matrix–vector products with matrices that are rational functions of S, which in turn require solving large, sparse linear systems. For large-scale discretization of the PDEs, practical implementations entail the use of preconditioned iterative methods for performing these solves. In addition, since many linear solves with a common matrix are required, recycling Krylov subspace methods (see Parks *et al.*, 2006) can also be applied to accelerate the solves. Our aim in this work is to explore the effectiveness of several algorithms for solving the sequence of linear systems arising from Lyapunov inverse iteration, and, more importantly, to develop a way of reducing the number of these solves.

The rest of this paper is organized as follows. In Section 2, we review two iterative Lyapunov solvers: the standard Krylov subspace method developed in Saad (1990) and the rational Krylov subspace method (RKSM) developed in Druskin & Simoncini (2011), both of which have been used in Elman & Wu (2013). In particular, we introduce the types of linear systems that need to be solved in the implementation of these two methods. In Section 3, we first review and test the iterative methods developed for these systems arising from incompressible Navier–Stokes equations. Then we incorporate these methods as well as Krylov subspace recycling into the Lyapunov solvers, which are tested on several examples considered. Based on the numerical results, we propose in Section 4 a modified version of the RKSM, and demonstrate that it achieves significant savings in computational cost. In Section 5, justification for this modification is provided. Some concluding remarks are given in Section 6.

2. Review of iterative Lyapunov solvers

In this section, we review two Lyapunov solvers that can be used for Lyapunov inverse iteration; see Elman & Wu (2013): the standard Krylov subspace method (Saad, 1990; Jaimoukha & Kasenally, 1994) and the RKSM (Druskin & Simoncini, 2011; Druskin *et al.*, 2011). Both methods construct a low-rank approximate solution to (1.4) of the form $Y_m = V_m X_m V_m^{\mathrm{T}}$, where the columns of V_m form an orthonormal basis for a small subspace of $\mathbb{R}^n$, and X_m is the solution to a small Lyapunov equation that can be solved using direct methods (Bartels & Stewart, 1972; Hammarling, 1982). Let the residual associated with (1.4) be

$$R = S(V_m X_m V_m^{\mathrm{T}}) + (V_m X_m V_m^{\mathrm{T}})S^{\mathrm{T}} - PCP^{\mathrm{T}}. \tag{2.1}$$

The small Lyapunov equation is obtained by imposing a Galerkin condition of the form $\mathrm{trace}(RZ^{\mathrm{T}}) = 0$, where Z is any matrix of the form $V_m Q V_m^{\mathrm{T}}$.

The right-hand side of the Lyapunov equation (1.4) that needs to be solved in the first step of Lyapunov inverse iteration is of rank 1, i.e., P is a single vector with unit norm (see Elman & Wu, 2013). As the algorithm proceeds, in subsequent iterations, the rank of the right-hand side of (1.4) may change to 2, in which case P will have two orthonormal columns. For simplicity, in the rest of this paper, we focus on the solution of the first Lyapunov equation. It is straightforward to generalize the algorithms presented here to the case where P has multiple columns.

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

$$\mathscr{K}_m(S, P) = \mathrm{span}\{P, SP, S^2P, \ldots, S^{m-1}P\},$$

is built. This method is outlined in Algorithm 2.1.

ALGORITHM 2.1 The standard Krylov subspace method for (1.4)

1. Given a tolerance τ. Let $v_1 = V_1 = P$.
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

$$H_m X_m + X_m H_m^T = \left(V_m^T P\right) C \left(V_m^T P\right)^T$$

   where $H_m = V_m^T S V_m$. ($Y_m = V_m X_m V_m^T$ is then the approximate solution to (1.4).)
   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, $V_{m+1} \leftarrow [V_m, v_{m+1}]$.

In Algorithm 2.1, the matrix $H_m = {V_m}^{\mathrm{T}} S V_m$ is available at no cost since it is simply the upper Hessenberg matrix $[h_{i,j}]_{i,j=1}^m$. As shown in Jaimoukha & Kasenally (1994), the residual norm $\|R\|_F$ of (1.4) can be computed cheaply as well. At each step of Algorithm 2.1, the matrix–vector product Sv_m is formed. Since $S = \mathbf{A}^{-1}\mathbf{M}$, where $\mathbf{A}$ is the Jacobian matrix, computing Sv_m entails one solve of the linear system

$$\mathbf{A}x = b, \tag{2.2}$$

where $b = \mathbf{M}v_m$. This is precisely the kind of linear system that needs to be solved in the computation of the steady-state solution for (1.1).

The RKSM was originally developed in Ruhe (1984, 1994) for the computation of the interior eigenvalues of S. This method constructs a subspace

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

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. The utility of this method for solving large-scale Lyapunov equations has recently been investigated in Druskin & Simoncini (2011). An algorithmic description reads as follows:

ALGORITHM 2.2 The RKSM for (1.4)

1. Given a tolerance τ and a shift s_1. Let $v_1 = V_1 = P$.
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 $T_m = V_m^T S V_m$ and solve the small Lyapunov equation

$$T_m X_m + X_m T_m^T = \left(V_m^T P\right) C \left(V_m^T P\right)^T.$$

($Y_m = V_m X_m V_m^T$ is then the approximate solution to (1.4).)
2.4. If $\|R\|_F < \tau$, then stop.
2.5. Else, $V_{m+1} \leftarrow [V_m, v_{m+1}]$ and compute the next shift s_{m+1}.

In Druskin *et al.* (2010) (for symmetric S) and Druskin & Simoncini (2011) (for general S), adaptive and parameter-free approaches for generating the shifts $\mathbf{s}$ were proposed. Both approaches require some knowledge of the spectrum of S. In Druskin & Simoncini (2011), the first shift s_1 is chosen to be a rough estimate of either $-\mathrm{Re}_{\min}(\theta)$ or $-\mathrm{Re}_{\max}(\theta)$, where $\mathrm{Re}_{\min}(\theta)$ and $\mathrm{Re}_{\max}(\theta)$ denote the minimum and maximum real parts of the eigenvalues of S, respectively. (Since we assume $\mathbf{A}$ to be the Jacobian matrix evaluated at any stable point $(\bar{u}_0, \alpha_0)$, the eigenvalues of S all lie in the left half of the complex plane, i.e., $0 > \mathrm{Re}_{\max}(\theta) > \mathrm{Re}_{\min}(\theta)$.) Let $\mathcal{I} = [-\mathrm{Re}_{\max}(\theta), -\mathrm{Re}_{\min}(\theta)]$ and $\{\hat{\theta}_j\}_{j=1}^m$ denote the eigenvalues of T_m, which will be updated at each iteration of Algorithm 2.2 and reflect the most recent information on the spectrum of S. Each subsequent shift s_{m+1} is then chosen as follows:

$$s_{m+1} = \arg\left(\max_{s\in\mathcal{I}} \frac{1}{|r_m(s)|}\right), \quad \text{where } r_m(s) = \frac{\prod_{j=1}^m (s - \hat{\theta}_j)}{\prod_{j=1}^m (s - s_j)}. \tag{2.3}$$

Once $\{\hat{\theta}_j\}_{j=1}^m$ are known, this selection process only involves sampling the rational function $r_m(s)$ on the interval $\mathcal{I}$, which is cheap. By Druskin & Simoncini (2011, Proposition 4.2), the residual norm $\|R\|_F$ of this method is also easy to compute.

We now look into the linear systems that arise from Algorithm 2.2 when it is applied to (1.4). Since

$$(S - s_m I)^{-1} = (\mathbf{A}^{-1}\mathbf{M} - s_m \mathbf{A}^{-1}\mathbf{A})^{-1} = (\mathbf{M} - s_m \mathbf{A})^{-1}\mathbf{A}, \tag{2.4}$$

computing $(S - s_m I)^{-1} v_m$ entails the solution of a linear system of the form

$$(\mathbf{M} - s\mathbf{A})x = b, \tag{2.5}$$

where $s > 0$ and $b = \mathbf{A}v_m$. The structure of (2.5) is exactly like that of the linear systems that need to be solved in the computation of a fully implicit iteration for a transient solution to (1.1), where s plays the role of the time step Δt.

Unlike in the standard Krylov subspace method, extra work is required to obtain the matrix $T_m = V_m^{\mathrm{T}} S V_m$ in step 2 of Algorithm 2.2. Computing it naïvely requires m matrix–vector products with S, i.e., m solves with $\mathbf{A}$. A more efficient way of generating this matrix was proposed in Ruhe (1994) (see also Druskin & Simoncini, 2011, Proposition 4.1), which only requires knowing Sv_{m+1}, or equivalently, one solve of (2.2) where $b = \mathbf{M}v_{m+1}$.

To sum up, when applied to (1.4), each iteration of the standard Krylov subspace method requires one solve of (2.2) in order to compute a new Krylov vector, whereas each iteration of RKSM requires one solve of (2.5) to compute a new Krylov vector and an additional solve of (2.2) to construct the matrix $T_m = V_m^{\mathrm{T}} S V_m$.

3. Numerical results

In the previous section, we have shown that solving the Lyapunov equation (1.4) arising from Lyapunov inverse iteration requires multiple solves of the linear systems (2.2) and/or (2.5), which are essentially the types of equations that arise in solving steady or transient PDEs. Therefore, iterative solution methods developed for such PDEs can be applied directly in Lyapunov inverse iteration. These strategies often involve designing efficient preconditioners that approximate the discrete versions of certain differential operators.

The dynamical system (1.1) that we consider in this study is the spatial discretization of the Navier–Stokes equations modelling incompressible flows,

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

subject to appropriate boundary conditions, where u, p, ν denote the velocity, pressure and kinematic viscosity, respectively. In order to examine the performance of Lyapunov solvers with iterative linear solves, we proceed in two steps: review some preconditioners developed for (3.1) and test them on (2.2) and (2.5), and then integrate these solution techniques into the Lyapunov solvers described in the previous section and apply them to (1.4) arising from Lyapunov inverse iteration.

3.1 *Iterative solves of the linear systems*

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

$$\mathbf{A} = \begin{bmatrix} F & B^{\mathrm{T}} \\ B & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{M} = \begin{bmatrix} -G & 0 \\ 0 & 0 \end{bmatrix}. \tag{3.2}$$

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 $n_p < n_u$. By G we denote 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 Elman *et al.* (2005).) Since a nonsingular mass matrix is required (see Elman & Wu, 2013), in the actual computation, we use instead a modified mass matrix

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

where $\eta = -0.01$. The rightmost eigenvalue of $\mathbf{A}x = \mu\mathbf{M}x$ remains unchanged when this new mass matrix is used (see Cliffe *et al.*, 1994).

Preconditioners for (2.2) of the form

$$\mathbf{P} = \begin{bmatrix} P_F & B^{\mathrm{T}} \\ 0 & -P_S \end{bmatrix} \tag{3.3}$$

were developed in Elman *et al.* (2005) and Elman (2005), where P_F is a preconditioner for F and P_S is a preconditioner for the (dense) Schur complement $BF^{-1}B^{\mathrm{T}}$. In (3.3), we can choose P_F to be F itself and apply its inverse using a multigrid process (see Elman *et al.*, 2005). The main issue in designing $\mathbf{P}$ is the choice of P_S.

In Elman *et al.* (2005), two effective preconditioning strategies for the Schur complement were described: the pressure convection–diffusion (PCD) preconditioner and the least-squares commutator (LSC) preconditioner. They are derived by approximately minimizing the discrete version of a commutator of certain differential operators using two different heuristics (see Elman *et al.*, 2005). The PCD preconditioner is defined to be

$$A_{\mathrm{p}}F_{\mathrm{p}}^{-1}G_{\mathrm{p}},$$

where A_{p} and F_{p} are discrete Laplacian and convection–diffusion operators in the pressure space, respectively, and G_{p} is the pressure mass matrix. The LSC preconditioner is defined to be

$$(B\hat{G}^{-1}B^{\mathrm{T}})(B\hat{G}^{-1}F\hat{G}^{-1}B^{\mathrm{T}})^{-1}(B\hat{G}^{-1}B^{\mathrm{T}}),$$

where $\hat{G}$ is the diagonal matrix whose entries are taken from the diagonal of G.

The coefficient matrix $\mathbf{M} - s\mathbf{A}$ of (2.5) has the block structure

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

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

$$-(\eta - s)^2 A_{\mathrm{p}}(sF_{\mathrm{p}} + G_{\mathrm{p}})^{-1}G_{\mathrm{p}} \tag{3.4}$$

and

$$-(\eta - s)^2 (B\hat{G}^{-1}B^{\mathrm{T}})(B\hat{G}^{-1}(sF+G)\hat{G}^{-1}B^{\mathrm{T}})^{-1}(B\hat{G}^{-1}B^{\mathrm{T}}), \tag{3.5}$$

respectively. The shift s in RKSM varies from one iteration to another. An important feature of both preconditioners is that they do not require any extra work to construct as s changes.

We will explore both these preconditioning strategies, using improved variants of them developed to account for boundary effects; see Elman & Tuminaro (2009) for details of these improvements.

Also, the description above is for 'ideal' versions of these methods. As described, their main computational costs are for linear system solves with F or $sF + G$ (where F resembles a convection–diffusion operator on the velocity space and G is a velocity mass matrix), A_{p} or $B\hat{G}^{-1}B^{\mathrm{T}}$ (pressure Poisson operators) and G_{p} (a pressure mass matrix). These subsidiary systems could be solved efficiently using multigrid for F or $sF + G$ and for A_{p}, but it is actually more effective to replace accurate solutions with approximate ones obtained, for example, by applying *a single step* of multigrid to the systems (see Elman *et al.*, 2005). In the results described below, P_F is defined using one V-cycle of algebraic multigrid (AMG) (implemented in the IFISS software package; see Elman *et al.*, 2007); the pressure Poisson solves are also replaced by a single V-cycle of AMG.[1] The pressure mass matrix is replaced by its diagonal approximation.

We investigate the utility of these preconditioning strategies for solving (2.2) and (2.5). The iterative linear solver we use for the preconditioned systems is GMRES (without restarts). Recall from Section 2 that we need to solve an instance of (2.2) in each iteration of both Algorithms 2.1 and 2.2, and an additional instance of (2.5) in each iteration of Algorithm 2.2. The right-hand side b in (2.2) and (2.5) and the shift in (2.5) vary from one iteration to another. In this subsection, as a simple first test, we

[1] On the velocity space, AMG is actually applied to the system with coefficient matrix $\hat{F}$ or $s\hat{F} + G$, where $\hat{F}$ is the block convection–diffusion operator that would be obtained from a Picard linearization of the Navier–Stokes operator; see Elman *et al.* (2005) for details.

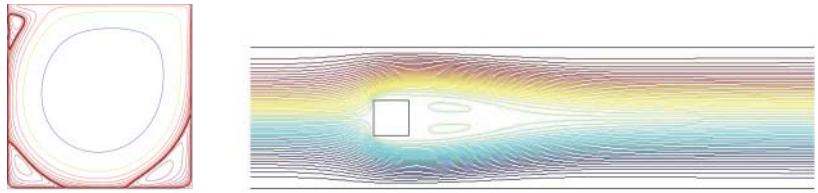

FIG. 1. The streamlines of driven-cavity flow (at $\mathscr{RE} = 4000$) and flow over an obstacle (at $\mathscr{RE} = 350$).

take b to be a constant vector with unit norm whose entries are all equal to $n^{-0.5}$ and consider five representative values of the shift, i.e., $s = 10^{-3}, 10^{-2}, 10^{-1}, 1, 10$. The stopping criteria for GMRES are

$$\|\mathbf{A}x - b\|_2 < 10^{-10} \cdot \|b\|_2 \tag{3.6}$$

for (2.2) and

$$\|(\mathbf{M} - s\mathbf{A})x - b\|_2 < 10^{-10} \cdot \|b\|_2 \tag{3.7}$$

for (2.5).

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

- two-dimensional flow over an obstacle (32×128 mesh, $n = 9512$) at $\mathscr{RE} = 200$ and $\mathscr{RE} = 350$, and
- two-dimensional driven-cavity flow (64×64 mesh, $n = 9539$) at $\mathscr{RE} = 2000$ and $\mathscr{RE} = 4000$.

Here, $\mathscr{RE}$ denotes the *Reynolds number* UL/ν, where U is the maximum magnitude of velocity on the inflow and L is a characteristic length scale for the domain. (In these tests, $L = 2$ and $U = 1$ for the obstacle problem, so that $\mathscr{RE} = 2/\nu$, and $L = 1$, $U = 1$, $\mathscr{RE} = 1/\nu$ for the cavity problem.) In all four examples, $n \approx 10{,}000$. Stability analysis of these two flows has been considered in previous work (Elman *et al.*, 2012; Elman & Wu, 2013), and their streamlines are depicted in Fig. 1. The critical Reynolds number is about 370 for the flow over an obstacle and approximately 8000 for the driven-cavity flow (see Elman *et al.*, 2012).

Figures 2 and 3 display the performance of the two preconditioners for solving (2.2) and (2.5) arising from these four examples. In each subplot, the residual norms $\|\mathbf{A}x - b\|_2$ and $\|(\mathbf{M} - s\mathbf{A})x - b\|_2$ are plotted against the number of preconditioned GMRES steps. The number next to each curve indicates the value of the shift s. The curves labelled with '∞' correspond to (2.2) since formally, (2.2) can be viewed as (2.5) with $s = \infty$.

The performance of GMRES for these representative shifts shows that the smaller the shift s is, the easier it is to solve (2.5). This is because the smaller s is, the better both preconditioners (3.4) and (3.5) approximate the Schur complement $-(\eta - s)^2 B(sF + G)^{-1}B^{\mathrm{T}}$. The number of GMRES steps needed by (2.2) shows the limit of how expensive solving (2.5) can get as s increases. It can also be seen that the performance of LSC is almost always stronger than that of PCD in these tests, and in the sequel we restrict our attention to LSC. The conclusions reached below for LSC would also apply in a qualitative sense to PCD.

From Figs 2 and 3, we can also observe that as the Reynolds number grows, when either preconditioner is used, (2.2) and (2.5) with a large shift become increasingly difficult to solve.

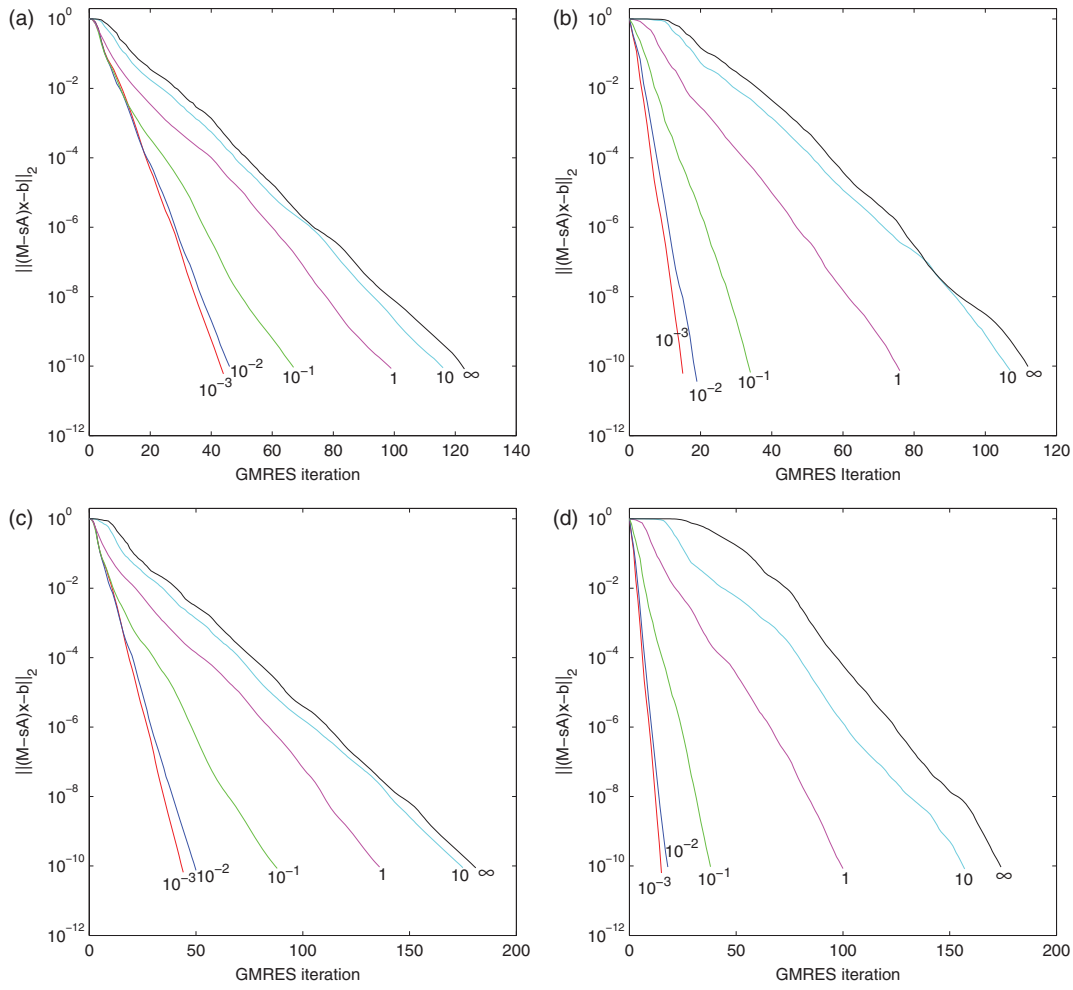

FIG. 2. The performance of the PCD preconditioner and the LSC preconditioner in flow over an obstacle. (a) PCD (Obstacle, $\mathscr{RE} = 200$), (b) LSC (Obstacle, $\mathscr{RE} = 200$), (c) PCD (Obstacle, $\mathscr{RE} = 350$), and (d) LSC (Obstacle, $\mathscr{RE} = 350$).

3.2 *Lyapunov solvers with iterative linear solves*

We next consider the performance of the full Lyapunov solvers when the linear solution methods described in Section 3.1 are integrated into the implementation.

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

$$\|R\|_F < 10^{-6} \cdot \|C\|_F \tag{3.8}$$

for the Lyapunov solve (outer iteration) and (3.6), (3.7) for the linear solves (inner iteration). All the linear systems arising from the Lyapunov solvers will be solved using GMRES in conjunction with the

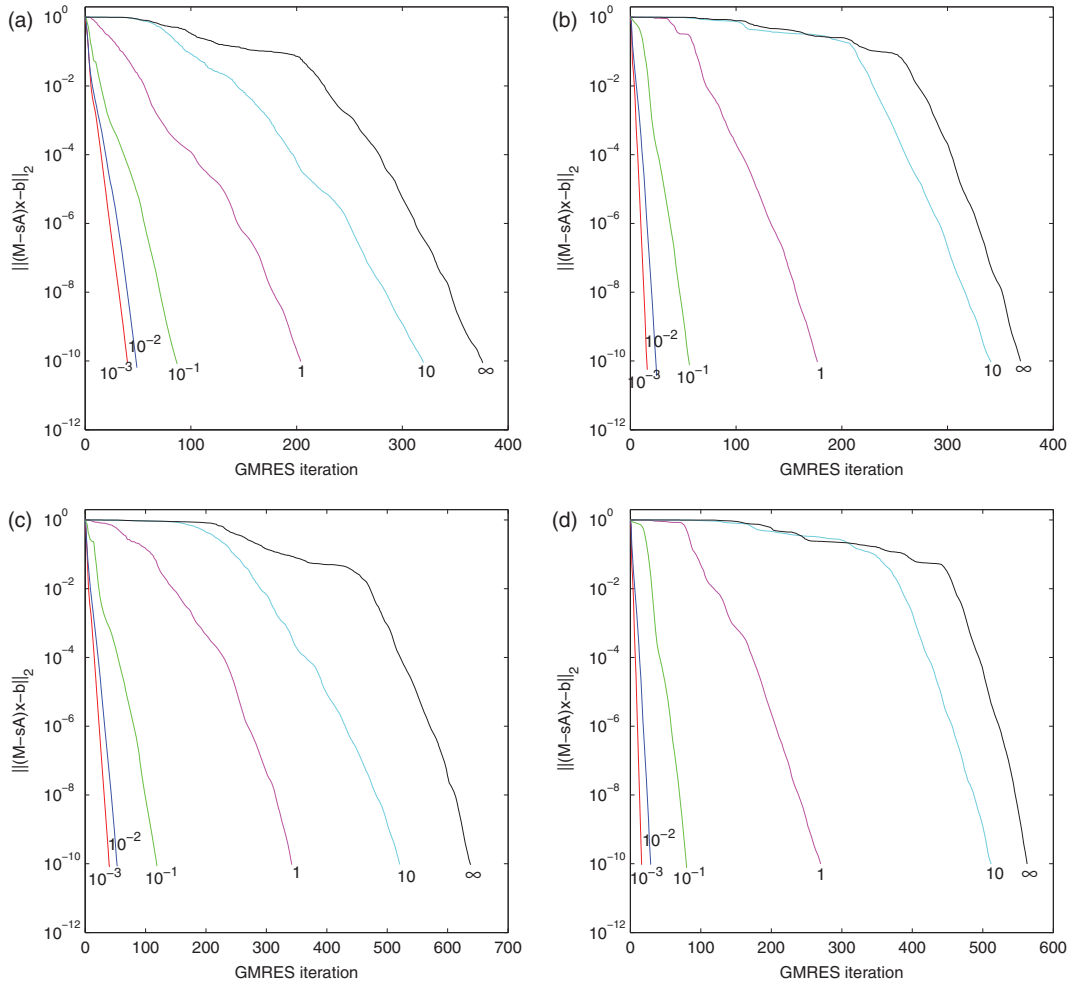

FIG. 3. The performance of the PCD preconditioner and the LSC preconditioner in driven-cavity flow. (a) PCD (Cavity, $\mathscr{RE} = 2000$), (b) LSC (Cavity, $\mathscr{RE} = 2000$), (c) PCD (Cavity, $\mathscr{RE} = 4000$), and (d) LSC (Cavity, $\mathscr{RE} = 4000$).

LSC preconditioner described in Section 3.1. In addition, as in the previous numerical experiments, the subsidiary linear systems arising from the application of $\mathbf{P}^{-1}$ to a vector are solved approximately using one multigrid V-cycle.

Recall that when applied to (1.4) where $S = \mathbf{A}^{-1}\mathbf{M}$, both Lyapunov solvers considered in Section 2 require solving a sequence of linear systems (2.2) with different right-hand sides, a typical scenario for which recycling Krylov subspaces (Parks *et al.*, 2006) can achieve significant computational savings. Suppose that preconditioned GMRES builds a Krylov subspace $\mathscr{K}_d^{(i)}(\mathbf{AP}^{-1}, r_i)$ for solving the ith instance of (2.2). The motivation behind recycling is that since all the linear systems in the sequence share the same coefficient matrix $\mathbf{A}$, the existing Krylov subspace $\mathscr{K}_d^{(i)}(\mathbf{AP}^{-1}, r_i)$ may contain certain information about the spectrum of $\mathbf{AP}^{-1}$ that will facilitate the solution of the $(i+1)$st system and, therefore, should not be discarded completely. We can retain d_r properly chosen linearly

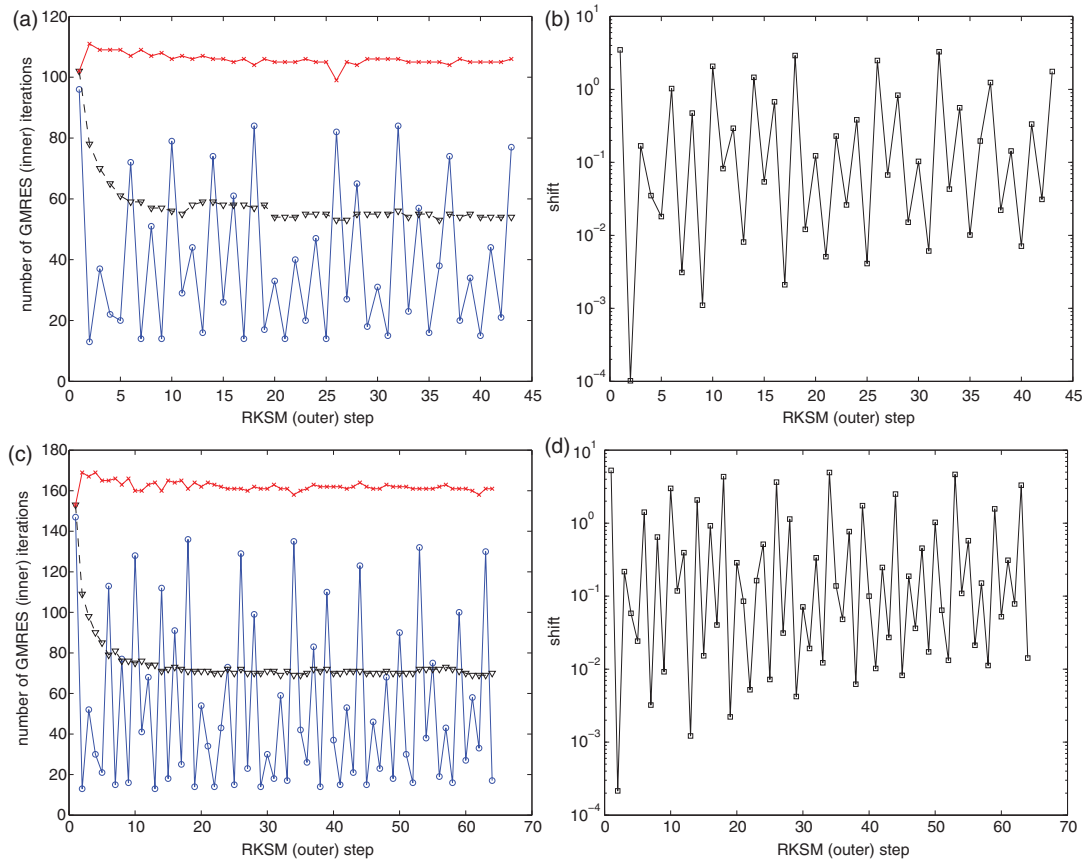

FIG. 4. The GMRES iteration count and shift in each step of the RKSM applied to the two obstacle cases (×: the number of GMRES iterations needed for solving (2.2), ▽: the number of GMRES iterations needed for solving (2.2) when recycling is used, ○: the number of GMRES iterations needed for solving (2.5), and □: the shift used in (2.5)). (a) GMRES iteration counts (Obstacle, $\mathscr{RE} = 200$), (b) Shifts (Obstacle, $\mathscr{RE} = 200$), (c) GMRES iteration counts (Obstacle, $\mathscr{RE} = 350$), and (d) Shifts (Obstacle, $\mathscr{RE} = 350$).

independent vectors in this subspace ($d_r < d$), and then augment them to form a new Krylov subspace $\mathscr{K}_d^{(i+1)}(\mathbf{AP}^{-1}, r_{i+1})$ for solving the $(i+1)$st instance of (2.2). If we construct such a subspace from scratch, d matrix–vector products with $\mathbf{AP}^{-1}$ are needed, whereas with recycling, only $d - d_r$ of them are required. As will be seen from the numerical experiments below, recycling improves the efficiency of both Lyapunov solvers considerably.

We first consider RKSM, which is shown in Algorithm 2.2. Besides the solve of (2.2), a solve of (2.5) is also needed at each iteration of this algorithm. The results of Algorithm 2.2 are shown in Figs 4 and 5, in which we plot both the number of GMRES steps (inner iterations) and the shift s for each iteration of RKSM (outer iteration).

In each of Figs 4(a,c), 5(a,c), there are three curves representing, respectively, the number of GMRES steps needed for solving (2.2) without recycling (denoted by '×'), the number of GMRES steps needed for solving the same system, but with recycling (denoted by '▽'), and the number of

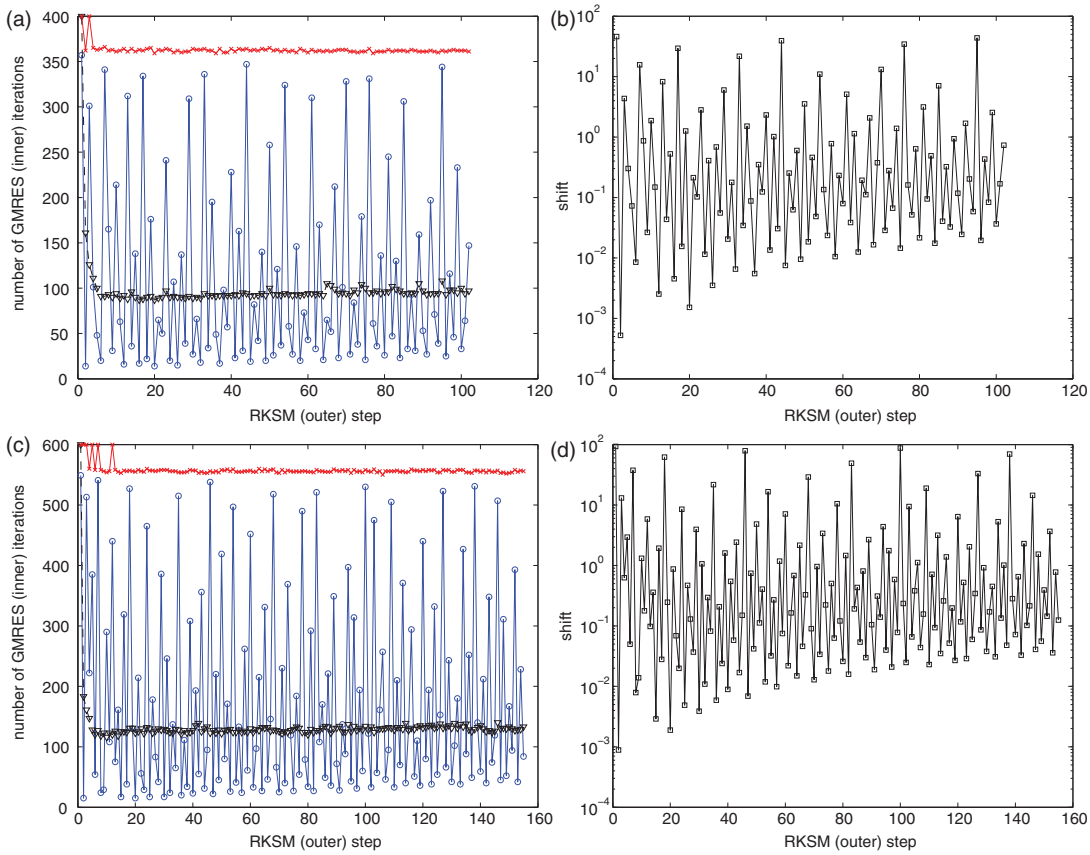

FIG. 5. The GMRES iteration count and shift in each step of the RKSM applied to the two cavity cases ($\times$: the number of GMRES iterations needed for solving (2.2), $\triangledown$: the number of GMRES iterations needed for solving (2.2) when recycling is used, $\circ$: the number of GMRES steps needed for solving (2.5), and $\square$: the shift used in (2.5)). (a) GMRES iteration counts (Cavity, $\mathscr{RE} = 2000$), (b) Shifts (Cavity, $\mathscr{RE} = 2000$), (c) GMRES iteration counts (Cavity, $\mathscr{RE} = 4000$), and (d) Shifts (Cavity, $\mathscr{RE} = 4000$).

GMRES steps for solving (2.5) (denoted by '$\circ$'), as Algorithm 2.2 proceeds.[2] As can be seen from these four figures, when no recycling is used, the number of GMRES steps needed for (2.2) is nearly a constant as the outer iteration advances. This constant is about 110 for $\mathscr{RE} = 200$, 170 for $\mathscr{RE} = 350$, 360 for $\mathscr{RE} = 2000$ and 560 for $\mathscr{RE} = 4000$, as given by the rightmost curve in Figs 2(b,d) and 3(b,d). With recycling, after a few outer (RKSM) steps, the number of recycled GMRES iterations is reduced considerably, to about 60, 70, 100 and 130 for these four cases. The improvements obtained from recycling are strongest for the systems arising from the cavity case with large Reynolds numbers, where (2.2) is rather difficult to solve. (Details of how the Krylov subspaces are recycled in these experiments are as follows. After solving (2.2) at the ith iteration of RKSM, we select from the Krylov subspace

[2] Since the shift s in (2.5) is often quite different from one iteration to the next, as demonstrated by Figs 4(b,d) and 5(b,d), the Krylov subspace built for solving one linear system may be of little help in the solution of the next one and, thus, recycling was not used for (2.5).

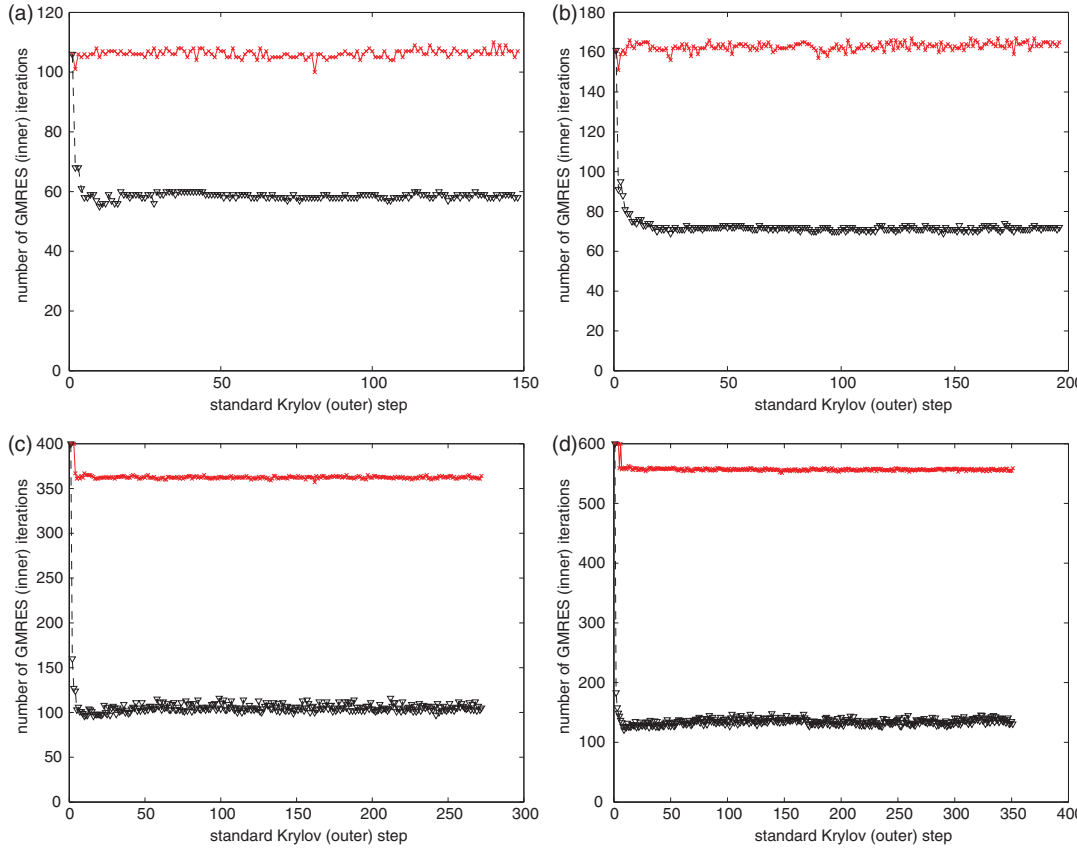

FIG. 6. The GMRES iteration count in each step of the standard Krylov subspace method ($\times$: the number of GMRES iterations needed for solving (2.2) and $\triangledown$: the number of GMRES iterations needed for solving (2.2) when recycling is used). (a) Obstacle, $\mathscr{RE} = 200$, (b) Obstacle, $\mathscr{RE} = 350$, (c) Cavity, $\mathscr{RE} = 2000$, and (d) Cavity, $\mathscr{RE} = 4000$.

$\mathscr{K}_d^{(i)}(\mathbf{AP}^{-1}, r_i)$ d_r harmonic Ritz vectors of $\mathbf{AP}^{-1}$ corresponding to its d_r smallest harmonic Ritz values, and will reuse them in the construction of the new Krylov subspace $\mathscr{K}_d^{(i+1)}(\mathbf{AP}^{-1}, r_{i+1})$ for solving the next instance of (2.2). In our experiments, d_r is chosen to be 60, 90, 200 and 300, respectively, for the four examples, which is roughly half of the number of GMRES steps required to solve (2.2) when no recycling is used.) From the same set of figures, it can be seen that the number of GMRES steps required to solve (2.5) is quite oscillatory and can change drastically from one iteration to the next. The pattern of oscillation matches perfectly with that of the shift, which is depicted in Figs 4(b,d), 5(b,d) (denoted by '□'). The bigger the shift is, the more GMRES steps are needed to solve (2.5). This behaviour is again expected from the numerical experiments in the previous section. In all four cases considered here, when no recycling is performed, approximately 75% of all the GMRES steps taken to solve the Lyapunov equation (1.4) are devoted to the solution of (2.2); recycling is able to cut this percentage to 40–60%, which is still large. In Section 4, we will present a technique that further reduces this part of the cost.

Next, we perform a similar test for the standard Krylov subspace method (Algorithm 2.1). We continue to use (3.6) as the stopping criterion for the linear solve (2.2) and (3.8) as the stopping criterion

for the Lyapunov solve. The numbers of GMRES iterations needed to solve each linear system arising from Algorithm 2.1, applied to the same set of examples, are shown in Fig. 6. Each subplot contains two curves representing, respectively, the number of GMRES iterations required to solve (2.2) with (denoted by '$\triangledown$') and without recycling (denoted by '$\times$'). (The way recycling is performed remains the same as in RKSM.) The behaviour of both sets of curves is very similar to that observed in Figs 4 and 5. Moreover, comparison between Figs 6 and 4, 5 shows that to solve (1.4) to the same order of accuracy, the standard Krylov subspace method needs a subspace more than twice as large as that needed by RKSM.

4. Modified RKSM

As shown in Section 3, compared with the standard Krylov method, RKSM can find an approximate solution to (1.4) with the same order of accuracy using a subspace of significantly smaller dimension. However, it requires solving both linear systems (2.2) and (2.5) at each iteration and, with or without recycling, a substantial portion of its total cost comes from solving (2.2). If the solution of (2.2) can somehow be avoided without harming the convergence rate of RKSM, then the efficiency of this method will increase significantly. In this section, we propose a modified version of RKSM for (1.4) that achieves this goal.

Recall from Section 2 that when RKSM is applied to (1.4), the linear system (2.5) arises from computing a new Krylov vector, and the linear system (2.2) arises from the computation of the matrix $T_m = \mathbf{V}^{\mathrm{T}} S \mathbf{V}$. We see that T_m is needed in the construction of the approximate solution $V_m X_m V_m^{\mathrm{T}}$ of (1.4), and its eigenvalues are used to generate the next shift s_{m+1}. However, it is not necessary to compute an approximate solution to (1.4) at each step of Algorithm 2.2; moreover, we only need the eigenvalues of T_m in (2.3), not T_m itself.

Thus, we can reduce the number of solves by not computing T_m at every step. We propose using the eigenvalues of $\mathscr{T}_m = (V_m^{\mathrm{T}} \mathbf{A} V_m)^{-1} (V_m^{\mathrm{T}} \mathbf{M} V_m)$ in (2.3) to generate the shift, instead of those of T_m. The reason is twofold: first, constructing $\mathscr{T}_m$ only requires matrix–vector products with $\mathbf{A}$ and $\mathbf{M}$; second, the eigenvalues of $\mathscr{T}_m$ approximate those of T_m well. The latter assertion is supported by numerical evidence given below and in Section 5.

Consider again (1.4) arising from driven-cavity flow at $\mathscr{RE} = 2000$. We compute the eigenvalues of both T_m and $\mathscr{T}_m$ as Algorithm 2.2 proceeds. For $m = 25, 50, 75, 100$, the spectra of T_m and $\mathscr{T}_m$ are plotted in Fig. 7, in which the crosses denote the eigenvalues of T_m and the circles represent the eigenvalues of $\mathscr{T}_m$. (Note that a logarithmic scale is used on the real axis for a clearer display of the eigenvalues.) As shown in Fig. 7, the eigenvalues of $\mathscr{T}_m$ indeed approximate those of T_m well, especially for larger m. This suggests that replacing the eigenvalues of T_m with those of $\mathscr{T}_m$ in the computation of the new shift will not affect the asymptotic convergence rate of RKSM.

The variant of RKSM with this modification is outlined in Algorithm 4.1. In the modified algorithm, we compute T_m and check the convergence of the Lyapunov solve only when the iteration count m is an integer multiple of a prescribed integer k. (When $k = 1$, this is simply Algorithm 2.2.) Consequently, (2.2) appears only in those iterations. In the iterations where T_m is not computed, we continue using the approach proposed in Druskin & Simoncini (2011) to choose the next shift; the only change is to use the eigenvalues of $\mathscr{T}_m$ instead of those of T_m in (2.3). The computation of $\mathscr{T}_m$ entails only two matrix–vector products $\mathbf{A}v_m$ and $\mathbf{M}v_m$ at each iteration. In fact, only the matrix–vector product $\mathbf{M}v_m$ is needed since, according to (2.4), $\mathbf{A}v_m$ is the right-hand side of (2.5) and has to be computed anyway. Thus, the cost of constructing $\mathscr{T}_m$ is negligible.

ALGORITHM 4.1 The modified RKSM for (1.4)

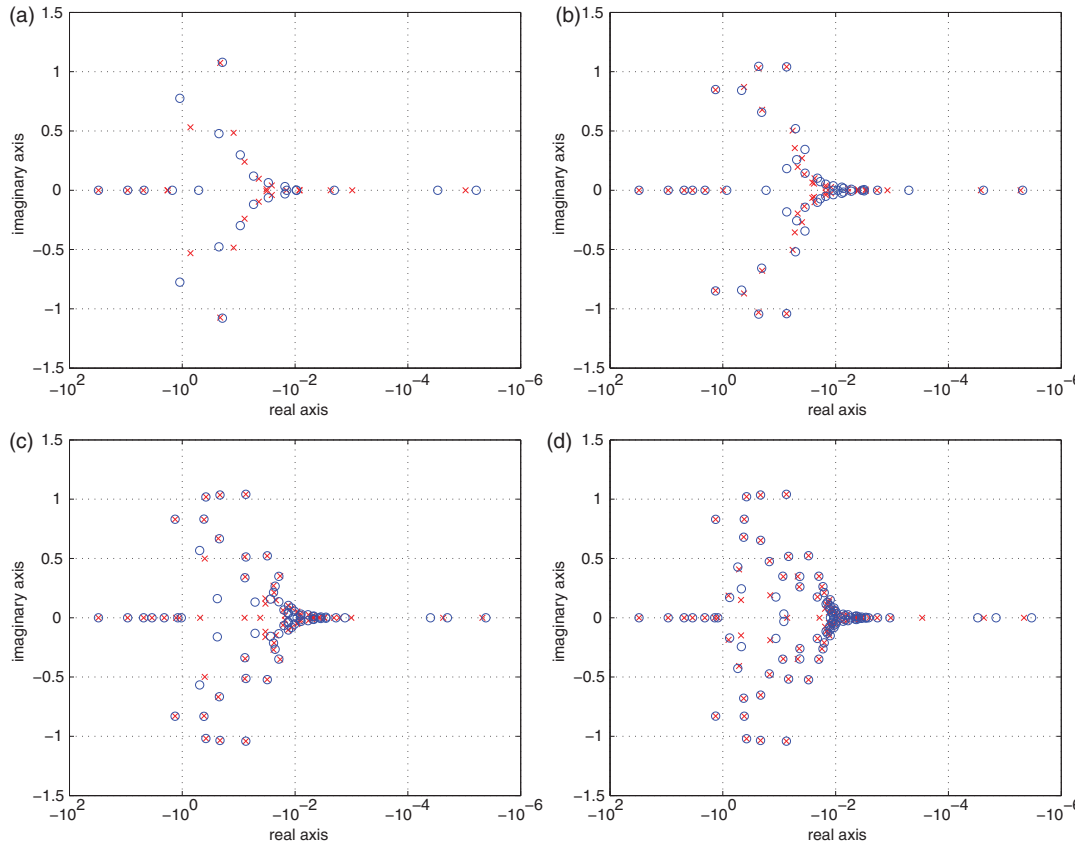

FIG. 7. The eigenvalues of T_m (crosses) and $\mathscr{T}_m$ (circles). (a) $m = 25$, (b) $m = 50$, (c) $m = 75$, and (d) $m = 100$.

1. Given a tolerance τ, a shift s_1 and an integer $k > 1$. Let $v_1 = V_1 = P$.
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(m,k) = 0**
   2.3.1. compute $T_m = V_m^T S V_m$ and solve the small Lyapunov equation

$$T_m X_m + X_m T_m^T = \left(V_m^T P\right) C \left(V_m^T P\right)^T;$$

   ($Y_m = V_m X_m V_m^T$ is then the approximate solution to (1.4).)
   2.3.2. if $\|R\|_F < \tau$, then stop.
   2.4. Else, compute $\mathscr{T}_m = (V_m^T \mathbf{A} V_m)^{-1} (V_m^T \mathbf{M} V_m)$.
   2.5. $V_{m+1} \leftarrow [V_m, v_{m+1}]$ and compute the next shift s_{m+1}.

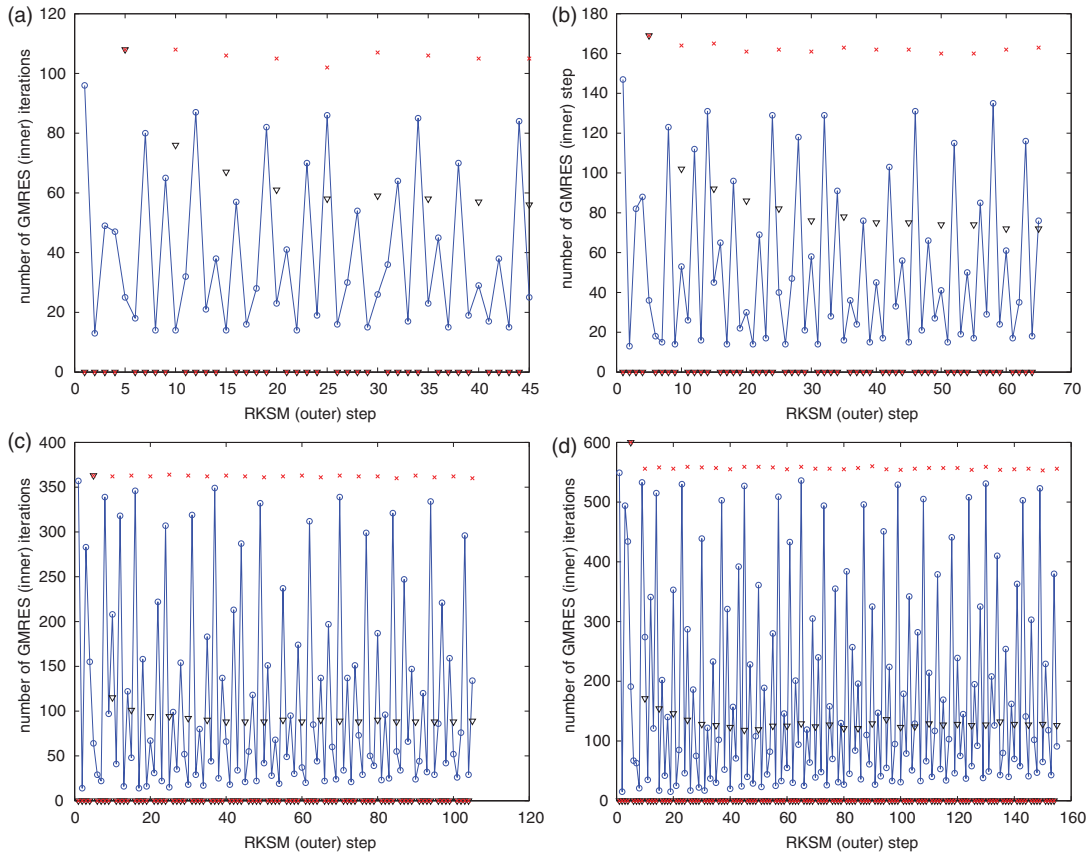

FIG. 8. The GMRES iteration count in each step of the modified RKSM with $k = 5$ ($\times$: the number of GMRES iterations needed for solving (2.2), $\triangledown$: the number of GMRES iterations needed for solving (2.2) when recycling is used, and $\circ$: the number of GMRES iterations needed for solving (2.5)). (a) Obstacle, $\mathscr{RE} = 200$, (b) Obstacle, $\mathscr{RE} = 350$, (c) Cavity, $\mathscr{RE} = 2000$, and (d) Cavity, $\mathscr{RE} = 4000$.

We apply Algorithm 4.1 (with $k = 5$) to the four examples considered in the previous section and display the numerical results in Fig. 8. We continue to use the stopping criteria (3.6), (3.7) for the inner iterations and (3.8) for the outer iterations. In Fig. 8, the triangle, the cross and the circle again denote the numbers of preconditioned GMRES steps required to solve (2.2) with and without recycling and (2.5), respectively. Since we only compute Sv_{m+1} every $k = 5$ iterations, as seen in Fig. 8, the number of GMRES iterations taken to solve (2.2) is simply zero in many iterations of Algorithm 4.1. By comparing Figs 4, 5 and 8, we also observe that, for the same example, the sizes of the Krylov subspaces built by Algorithm 2.2 and that built by Algorithm 4.1 are almost the same. (A slight increase (less than k) in the size of the Krylov subspace can be observed when Algorithm 4.1 is used instead of Algorithm 2.2. This is due to the fact that, in Algorithm 4.1, we only check convergence every k iterations and, therefore, the algorithm may not terminate even if (3.8) has already been met.) This implies that the shifts generated using the eigenvalues of $\mathscr{T}_m$ are essentially of the same quality as those generated using the eigenvalues of T_m.

Figures 9 and 10 provide a summary statement of the results obtained in this section and the previous one. The figures plot the residual norm $\|R\|_F$ associated with (1.4) against the total number of GMRES

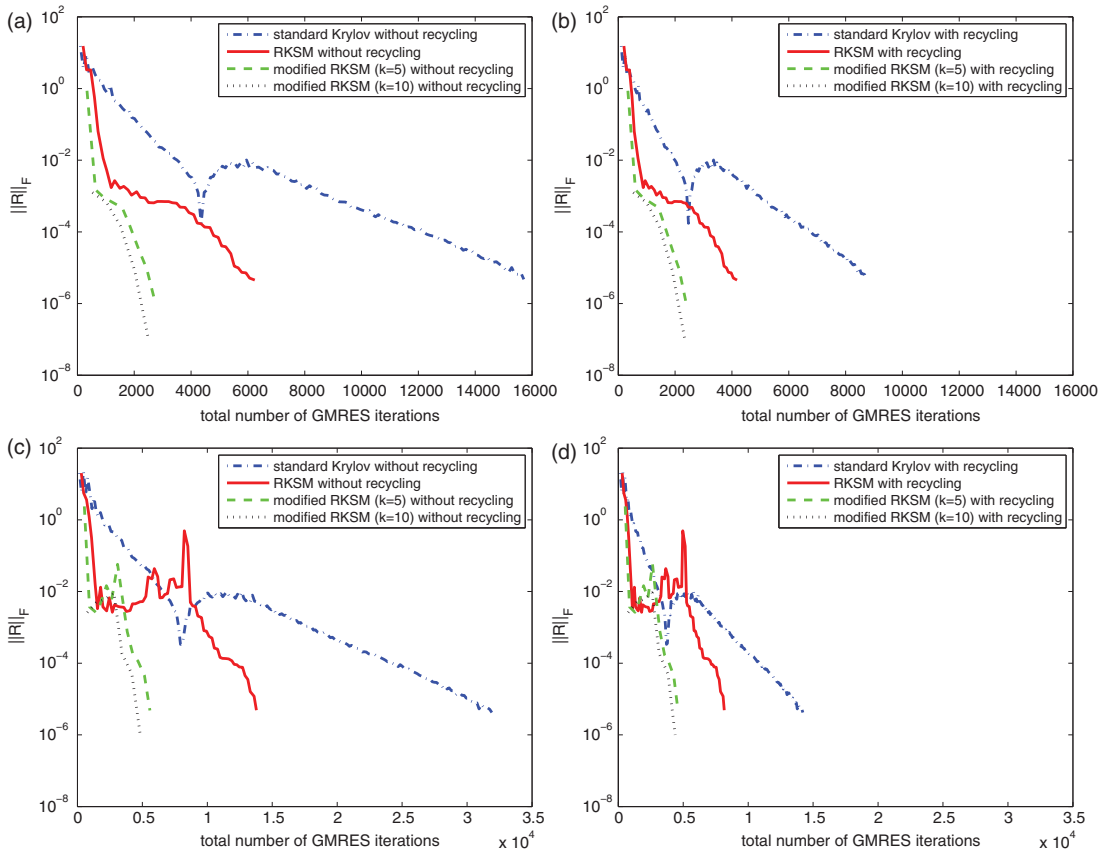

FIG. 9. Comparison of the total number of GMRES iterations required by the standard Krylov subspace method, RKSM and modified RKSM applied to the two obstacle cases. (a) Without recycling (Obstacle, $\mathscr{RE} = 200$), (b) With recycling (Obstacle, $\mathscr{RE} = 200$), (c) Without recycling (Obstacle, $\mathscr{RE} = 350$), and (d) With recycling (Obstacle, $\mathscr{RE} = 350$).

steps required by the standard Krylov subspace method (Algorithm 2.1), RKSM (Algorithm 2.2) and the modified RKSM (Algorithm 4.1) with $k = 5$ and 10. Figure 9 shows results for the obstacle problem and Fig. 10 for the driven-cavity problem. In both figures, performance without recycling is shown on the left and with recycling on the right. It can be seen that when no recycling is applied in solving (2.2), RKSM is much more efficient, requiring approximately half as many GMRES steps as the standard Krylov subspace method in all four examples. There are two reasons for this: first, RKSM requires a much smaller Krylov subspace, and second, although an extra solve of (2.5) is needed per iteration of RKSM, it is on average much cheaper to solve this system than to solve (2.2) without recycling (see Figs 4 and 5). When recycling is incorporated, RKSM still outperforms the standard Krylov method in three out of the four cases, though its superiority is less pronounced. By eliminating many of the system solves for (2.2), the modified version of RKSM, with or without recycling, reduces costs significantly in all four cases.

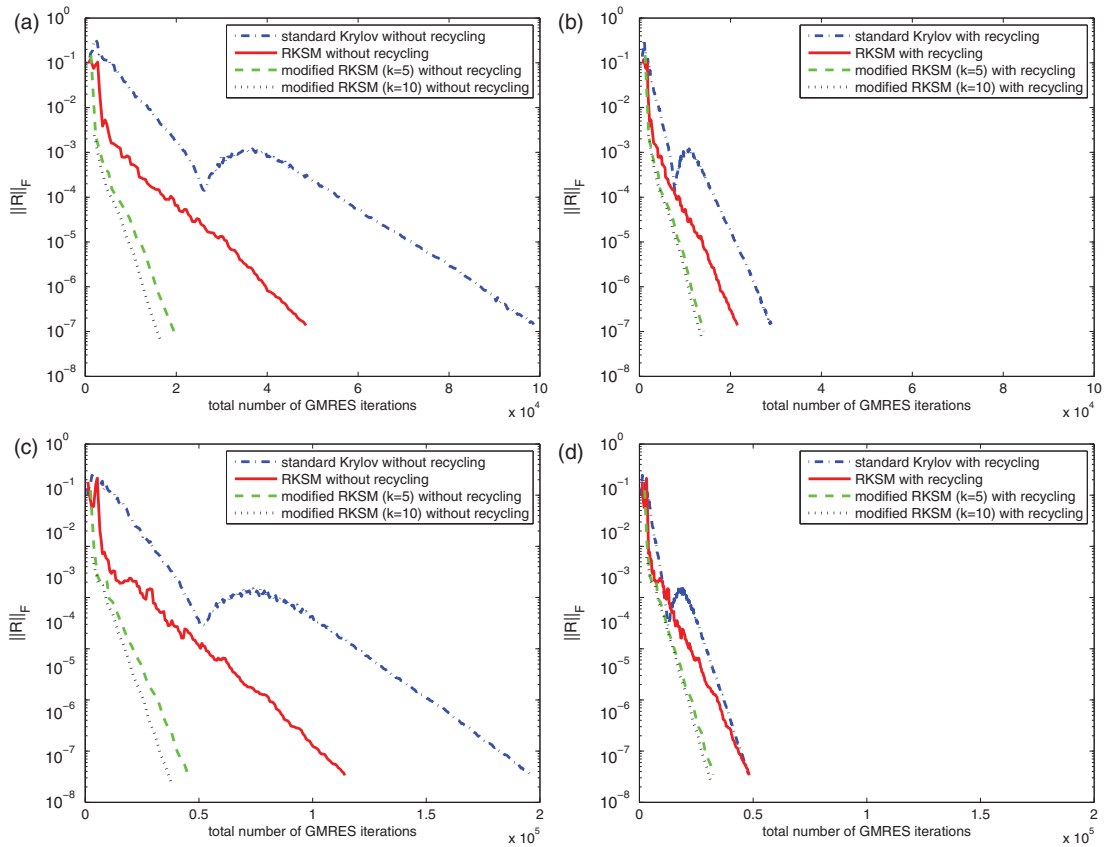

FIG. 10. Comparison of the total number of GMRES iterations required by the standard Krylov subspace method, RKSM and modified RKSM applied to the two cavity cases. (a) Without recycling (Cavity, $\mathscr{RE} = 2000$), (b) With recycling (Cavity, $\mathscr{RE} = 2000$), (c) Without recycling (Cavity, $\mathscr{RE} = 4000$), and (d) With recycling (Cavity, $\mathscr{RE} = 4000$).

5. Further analysis of T_m and $\mathscr{T}_m$

The modified RKSM (Algorithm 4.1) proposed in Section 4 is based on the hypothesis that the eigenvalues of $\mathscr{T}_m = (V_m^{\mathrm{T}}\mathbf{A}V_m)^{-1}(V_m^{\mathrm{T}}\mathbf{M}V_m)$ approximate those of $T_m = V_m^{\mathrm{T}}SV_m = V_m^{\mathrm{T}}(\mathbf{A}^{-1}\mathbf{M})V_m$ well. In this section, we analyse the relation between $\mathscr{T}_m$ and T_m, and provide more numerical evidence to support this hypothesis.

As shown in Ruhe (1994) (also see the Druskin & Simoncini, 2011, proof of Proposition 4.2), the RKSM computes the Arnoldi decomposition

$$SV_m = V_m T_m + v_{m+1}h_{m+1,m}e_m^{\mathrm{T}}D_m H_m^{-1} - (I - V_m V_m^{\mathrm{T}})Sv_{m+1}h_{m+1,m}e_m^{\mathrm{T}}H_m^{-1} \tag{5.1}$$

at each iteration, where e_m is the last column of the $m \times m$ identity matrix, $D_m = \mathrm{diag}(\{s_1, s_2, \ldots, s_m\})$ is the diagonal matrix that holds all the previous shifts, and I is the $n \times n$ identity matrix. The definitions of V_m, H_m, T_m, $h_{m+1,m}$ and v_{m+1} can be found in Algorithm 2.2.

TABLE 1 $\|E_m\|_2/\|T_m\|_2$

m	Obstacle $\mathscr{RE} = 200$	Obstacle $\mathscr{RE} = 350$	Cavity $\mathscr{RE} = 2000$	Cavity $\mathscr{RE} = 4000$
25	0.25732	0.25663	0.03824	0.01553
50	0.14088	0.10365	0.03453	0.02342
75	0.05734	0.08111	0.02904	0.01784
100	0.04143	0.05611	0.01797	0.02037

THEOREM 1 $\mathscr{T}_m = T_m + E_m$ where E_m is of rank 1. The single nonzero eigenvalue of E_m is

$$\rho_m = h_{m+1,m} e_m^{\mathrm{T}} H_m^{-1} (V_m^{\mathrm{T}} \mathbf{A} V_m)^{-1} V_m^{\mathrm{T}} \mathbf{A} (I - V_m V_m^{\mathrm{T}})(s_m I - S) v_{m+1},$$

and the corresponding eigenvector is

$$\psi_m = (V_m^{\mathrm{T}} \mathbf{A} V_m)^{-1} V_m^{\mathrm{T}} \mathbf{A} (I - V_m V_m^{\mathrm{T}})(s_m I - S) v_{m+1}.$$

Proof. Left multiply both sides of (5.1) by $V_m^{\mathrm{T}}\mathbf{A}$:

$$\begin{aligned} V_m^{\mathrm{T}} \mathbf{M} V_m = {} & (V_m^{\mathrm{T}} \mathbf{A} V_m) T_m + V_m^{\mathrm{T}} \mathbf{A} v_{m+1} h_{m+1,m} e_m^{\mathrm{T}} D_m H_m^{-1} \\ & - V_m^{\mathrm{T}} \mathbf{A} (I - V_m V_m^{\mathrm{T}}) S v_{m+1} h_{m+1,m} e_m^{\mathrm{T}} H_m^{-1}. \end{aligned} \tag{5.2}$$

Then, left multiply both sides of (5.2) by $(V_m^{\mathrm{T}} \mathbf{A} V_m)^{-1}$:

$$\begin{aligned} \mathscr{T}_m = {} & T_m + (V_m^{\mathrm{T}} \mathbf{A} V_m)^{-1} V_m^{\mathrm{T}} \mathbf{A} v_{m+1} h_{m+1,m} e_m^{\mathrm{T}} D_m H_m^{-1} \\ & - (V_m^{\mathrm{T}} \mathbf{A} V_m)^{-1} V_m^{\mathrm{T}} \mathbf{A} (I - V_m V_m^{\mathrm{T}}) S v_{m+1} h_{m+1,m} e_m^{\mathrm{T}} H_m^{-1}. \end{aligned} \tag{5.3}$$

Since $e_m^{\mathrm{T}} D_m = s_m e_m^{\mathrm{T}}$ and v_{m+1} is orthogonal to the columns of V_m, the difference between $\mathscr{T}_m$ and T_m is

$$E_m = h_{m+1,m} (V_m^{\mathrm{T}} \mathbf{A} V_m)^{-1} V_m^{\mathrm{T}} \mathbf{A} (I - V_m V_m^{\mathrm{T}})(s_m I - S) v_{m+1} e_m^{\mathrm{T}} H_m^{-1}.$$

It is easy to check that E_m has $m - 1$ zero eigenvalues whose eigenvectors are given by $\{\psi_j = H_m e_j\}_{j=1}^{m-1}$, and the single nonzero eigenvalue ρ_m whose corresponding eigenvector is ψ_m. □

Thus, $\mathscr{T}_m$ differs from T_m by a matrix of rank 1. We have seen that the eigenvalues of these two matrices are very close to each other. In Table 1, we also report the 'relative error' $\|E_m\|_2/\|T_m\|_2$ for several different values of m and each of the four examples considered in Sections 3 and 4. It can be seen that these errors are small, giving further indication that $\mathscr{T}_m$ is close to T_m.

6. Conclusions

In this paper, we explore the performance of the standard Krylov subspace method and the RKSM 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: one with structure identical to those that arise in the computation of steady states of a system of PDEs, and one 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 the RKSM. One way of cutting this cost is to

recycle the Krylov subspaces built for solving these systems, which is shown to be very effective for all our examples, especially the ones that are challenging to begin with. To further reduce this cost, we modify the RKSM 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.

Funding

This material is based upon work supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under Award Number DE-SC-0009301 and by the U.S. National Science foundation under grant DMS1115317.

References

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

Bartels, R. H. & Stewart, G. W. (1972) Algorithm 432: solution of the matrix equation $AX + XB = C$. *Commun. ACM*, **15**, 820–826.

Cliffe, K. A., Garratt, T. J. & Spence, A. (1994) Eigenvalues of block matrices arising from problems in fluid mechanics. *SIAM J. Matrix Anal. Appl.*, **15**, 1310–1318.

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

Druskin, V., Lieberman, C. & Zaslavsky, M. (2010) On adaptive choice of shifts in rational Krylov subspace reduction of evolutionary problems. *SIAM J. Sci. Comput.*, **32**, 2485–2496.

Druskin, V. & Simoncini, V. (2011) Adaptive rational Krylov subspaces for large-scale dynamical systems. *Systems Control Lett.*, **60**, 546–560.

Elman, H. C. (2005) Preconditioning strategies for models of incompressible flow. *J. Sci. Comput.*, **25**, 347–366.

Elman, H. C., Meerbergen, K., Spence, A. & Wu, M. (2012) Lyapunov inverse iteration for identifying Hopf bifurcations in models of incompressible flow. *SIAM J. Sci. Comput.*, **34**, A1584–A1606.

Elman, H. C., Ramage, A. & Silvester, D. J. (2007) Algorithm 866: IFISS, a Matlab toolbox for modelling incompressible flow. *ACM Trans. Math. Softw.*, **33**, 141–148.

Elman, H., Silvester, D. & Wathen, A. (2005) *Finite Elements and Fast Iterative Solvers*. Oxford: Oxford University Press.

Elman, H. C. & Tuminaro, R. S. (2009) Boundary conditions in approximate commutator preconditioners for the Navier–Stokes equations. *Electron. Trans. Numer. Anal.*, **35**, 257–280.

Elman, H. C. & Wu, M. (2013) Lyapunov inverse iteration for computing a few rightmost eigenvalues of large generalized eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, **34**, 1685–1707.

Grasedyck, L. (2004) Low-rank solutions of the Sylvester equation. *Numer. Linear Algebra Appl.*, **11**, 371–389.

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

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

Kressner, D. & Tobler, C. (2010) Low-rank tensor Krylov subspace methods for parameterized linear systems. *Technical Report* 2010-16. Zurich: ETH. Available from http://www.math.ethz.ch/ kressner/.

Meerbergen, K. & Spence, A. (2010) 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.

Meerbergen, K. & Vandebril, R. (2012) A reflection on the implicitly restarted Arnoldi method for computing eigenvalues near a vertical line. *Linear Algebra Appl.*, **436**, 2828–2844.

PARKS, M. L., DE STURLER, E., MACKEY, G., JOHNSON, D. D. & MAITI, S. (2006) Recycling Krylov subspaces for sequences of linear systems. *SIAM J. Sci. Comput.*, **28**, 1651–1674.

PENZL, T. (2000) Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case. *Systems Control Lett.*, **40**, 139–144.

RUHE, A. (1984) Rational Krylov sequence methods for eigenvalue computation. *Linear Algebr. Appl.*, **58**, 391–405.

RUHE, A. (1994) The rational Krylov algorithm for nonsymmetric eigenvalue problems. III: complex shifts for real matrices. *BIT*, **34**, 165–176.

SAAD, Y. (1990) Numerical solution of large Lyapunov equations. *Signal Processing, Scattering, Operator Theory, and Numerical Methods* (M. A. Kaashoek, J. H. van Schuppen & A. C. Ran eds). Proceedings of the International Symposium MTN-89, vol. III. Boston: Birkhauser, pp. 503–511.

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

STEWART, G. W. (2001) *Matrix Algorithms Volume II: Eigensystems*. Philadelphia: SIAM.