Noname manuscript No.
(will be inserted by the editor)

Methods for solution of large optimal control problems that bypass open-loop model reduction

Thomas Bewley · Paolo Luchini · Jan Pralits

Received: date / Accepted: date

Abstract Three algorithms for efficient solution of optimal control problems for high-dimensional systems are presented. Each bypasses the intermediate (and, unnecessary) step of open-loop model reduction. Each also bypasses the solution of the full Riccati equation corresponding to the LQR problem, which is numerically intractable for large n. Motivation for this effort comes from the field of model-based flow control, where open-loop model reduction often fails to capture the dynamics of interest (governed by the Navier-Stokes equation). Our *Minimum Control Energy* (MCE) method is a simplified expression for the well-known minimum-energy stabilizing control feedback that depends only on the left eigenvectors corresponding to the unstable eigenvalues of the system matrix A. Our *Adjoint of the Direct-Adjoint* (ADA) method is based on the repeated iterative computation of the adjoint of a forward problem, itself defined to be the direct-adjoint vector pair associated with the LQR problem. Our *Oppositely-Shifted Subspace Iteration* (OSSI, the main new result of the present paper) method is based on our new subspace iteration method for computing the Schur vectors corresponding, notably, to the $m \ll n$ central eigenvalues (near the imaginary axis) of the Hamiltonian matrix related to the Riccati equation of interest. Prototype OSSI implementations are tested on a low-order control problem to illustrate its behavior.

Thomas Bewley
Dept of MAE, UC San Diego, La Jolla CA 92093-0411, USA
E-mail: bewley@eng.ucsd.edu

Paolo Luchini
Università di Salerno, DIIN, 84084 Fisciano, Italy
E-mail: luchini@unisa.it

Jan Pralits
Università di Genova, DICCA, 16145 Genova, Italy
E-mail: jan.pralits@unige.it

Keywords Computational mechanics · Optimal control · minimum-energy control · subspace iteration

1 Introduction & Background

A primary difficulty of the linear feedback control problem is that its computational complexity scales poorly with problem size. Though it is quite routine with modern computers to perform numerical simulations of complex systems with state dimension $n \geq O(10^6)$, it is rare to see a feedback control problem solved directly on systems with state dimension larger than $O(10^3)$. Instead, the most common strategy is a two-step approach: first apply some sort of "balanced" open-loop model reduction to the system [16, 19], then solve a control problem based on this reduced-order model. While this two-step approach proves to be successful for some problems, it is problematical for others.

The reason for this difficulty is twofold. First, though a balanced truncation of a system accounts for the controllability and observability of the various eigenmodes of the system (i.e., the matrices B and C in the system model), such a truncation is inherently an "open-loop" model reduction strategy, and does not account for the closed-loop control objective (i.e., the matrices Q and R in the cost function).

The second issue is related to the condition of *eigenvector nonorthogonality* of the system matrix A. In systems characterized by this condition, the eigenvalues of the system matrix, and the controllability and observability of the corresponding eigenvectors, do not tell the whole story, and very large transfer-function norms and transient energy growth (a.k.a. "peaking") are possible even if all of the eigenvalues of the system are stable and "well damped". The mechanism for such transient energy growth is the possibility of initial *destructive interference* of multiple nonorthogonal eigenvectors of the system [6,4]; this destructive interference can reduce substantially in time (as different modes decay at different rates), leading to substantial energy *growth* in the system, before ultimate energy decay due to the stability of the corresponding eigenvalues. The energy growth via such mechanisms can be several orders of magnitude, and can thus lead quickly to nonlinear (a.k.a. "secondary") instability even when the initial perturbations on the system are quite small. Eigenvector nonorthogonality thus reduces the relevance of eigenmodes considered on their own, and model reduction strategies based on retaining certain open-loop eigenmodes from the spectrum, but not others, can lead to significant problems, as the full set of eigenvectors necessary to capture the transient energy growth mechanisms present in the system are generally not contained by a model that has been reduced in such a fashion [4,13].

Finally, the two-step approach described above appears to be an unfortunate duplication of effort: an algorithm with the complexity of an eigenvalue problem is first used to reduce the order of a model, then another algorithm with the complexity of an eigenvalue problem is used to solve a control problem based on this reduced-order model. The central idea of the present paper is to consider control formulations which solve one such eigenvalue problem, not two.

1.1 Brief review of the optimal control problem

As a point of reference for the derivations in the sections to come, it is necessary to review concisely the key equations and standard methods of solution of the optimal control problem (written here in its simplest form, which is sufficient for the discussion that follows). This background material is broadly known [5,16], and is leveraged heavily in the following form in the sections that follow. In short, we seek to minimize a finite-horizon cost function

$$J(\mathbf{x}(\mathbf{u}),\mathbf{u}) = \frac{1}{2}\int_0^T (\mathbf{x}^H Q\mathbf{x} + \mathbf{u}^H R\mathbf{u})\,dt + \frac{1}{2}\,\mathbf{x}^H(T)Q_T\mathbf{x}(T) \tag{1}$$

where $Q \geq 0$, $R > 0$, $Q_T \geq 0$, and $(\cdot)^H$ denotes the conjugate transpose, and where the state $\mathbf{x} = \mathbf{x}(\mathbf{u})$ is related to the control $\mathbf{u}$ via a linear (or, linearized) time-varying (LTV), possibly complex system

$$d\mathbf{x}/dt = A\mathbf{x} + B\mathbf{u} \quad \text{with} \quad \mathbf{x}(0) = \mathbf{x}_0. \tag{2a}$$

Toward this end, we may define an adjoint field related to the optimization problem of interest,

$$-d\mathbf{p}/dt = A^H\mathbf{p} + Q\mathbf{x} \quad \text{with} \quad \mathbf{p}(T) = Q_T\mathbf{x}(T). \tag{2b}$$

For any initial condition $\mathbf{x}_0$, the control $\mathbf{u}$ on $t \in [0,T]$ which minimizes $J(\mathbf{x}(\mathbf{u}),\mathbf{u})$ may be found using (2a)-(2b) by iterative state/adjoint computation, starting from an initial guess $\mathbf{u} = 0$ on $t \in [0,T]$ and at each iteration updating the control $\mathbf{u}$ (using, e.g., a conjugate gradient or BFGS method) based on the gradient

$$DJ(\mathbf{x}(\mathbf{u}),\mathbf{u})/D\mathbf{u} = B^H\mathbf{p} + R\mathbf{u}, \tag{2c}$$

computed using the adjoint field $\mathbf{p}$.

To verify the relevance of the adjoint equation (2b) for the minimization of $J(\mathbf{x}(\mathbf{u}),\mathbf{u})$ in (1) when $\mathbf{x}$ is related to $\mathbf{u}$ by (2a), consider a linear perturbation analysis of (1) and (2a): replacing $\mathbf{u} \leftarrow \mathbf{u} + \mathbf{u}'$, $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{x}'$, and $J \leftarrow J + J'$ and keeping all terms which are linear in the perturbation quantities gives

$$J' = \int_0^T (\mathbf{x}^H Q\mathbf{x}' + \mathbf{u}^H R\mathbf{u}')\,dt + \mathbf{x}^H(T)Q_T\mathbf{x}'(T) \tag{3a}$$

and a linear evolution equation for $\mathbf{x}'$:

$$\mathcal{L}\mathbf{x}' = B\mathbf{u}' \quad \text{with} \quad \mathbf{x}'(0) = 0 \tag{3b}$$

$$\text{where} \quad \mathcal{L} \triangleq d/dt - A. \tag{3c}$$

Defining the inner product $\langle \mathbf{a},\mathbf{b}\rangle = \int_0^T \mathbf{a}^H\mathbf{b}\,dt$, we may express an *adjoint identity*

$$\langle \mathbf{p}, \mathcal{L}\mathbf{x}'\rangle = \langle \mathcal{L}^*\mathbf{p}, \mathbf{x}'\rangle + b. \tag{3d}$$

Using integration by parts, it follows that

$$\mathcal{L}^* = -d/dt - A^H, \quad b = \mathbf{p}^H\mathbf{x}'|_{t=T} - \mathbf{p}^H\mathbf{x}'|_{t=0}. \tag{3e}$$

Using $\mathcal{L}^*$ to define an appropriate evolution equation for $\mathbf{p}$ [which is equivalent to (2b)],

$$\mathcal{L}^*\mathbf{p} = Q\mathbf{x} \quad \text{with} \quad \mathbf{p}(T) = Q_T\mathbf{x}(T), \tag{3f}$$

and substituting both (3b) and (3f) into the identity (3d) allows us to rewrite (3a) as

$$J' = \int_0^T \Big[B^H\mathbf{p} + R\mathbf{u}\Big]^H \mathbf{u}'\,dt = \int_0^T \Big[\frac{DJ}{D\mathbf{u}}\Big]^H \mathbf{u}'\,dt, \tag{3g}$$

from which the gradient in (2c) is readily identified.

Rather than using an iterative vector-based method to find the $\mathbf{u}$ on $t \in [0,T]$ to minimize $J(\mathbf{x}(\mathbf{u}),\mathbf{u})$ for the initial condition $\mathbf{x}_0$, leveraging (2a)-(2c) as described above, we may instead enforce the condition that $DJ(\mathbf{x}(\mathbf{u}),\mathbf{u})/D\mathbf{u} = 0$ directly, thus reducing (2c) to

$$\mathbf{u} = -R^{-1}B^H\mathbf{p}. \tag{4}$$

Substituting this condition into (2a) allows us to rewrite the two-point boundary-value problem (TPBVP) for the state/adjoint pair $\{\mathbf{x}, \mathbf{p}\}$, as listed in (2a)-(2b), in the convenient combined matrix form

$$\frac{d}{dt}\underbrace{\begin{bmatrix}\mathbf{x}\\ \mathbf{p}\end{bmatrix}}_{\mathbf{v}} = \underbrace{\begin{bmatrix} A & -BR^{-1}B^H \\ -Q & -A^H\end{bmatrix}}_{Z}\underbrace{\begin{bmatrix}\mathbf{x}\\ \mathbf{p}\end{bmatrix}}_{\mathbf{v}} \tag{5a}$$

with initial and terminal conditions

$$\begin{cases}\mathbf{x} = \mathbf{x}_0 & \text{at } t = 0,\\ \mathbf{p} = Q_T\mathbf{x} & \text{at } t = T,\end{cases} \tag{5b}$$

which may be solved for arbitrary initial conditions $\mathbf{x}_0$ by the sweep method: assuming a relation exists between the state vector $\mathbf{x}$ and adjoint vector $\mathbf{p}$ such that

$$\mathbf{p} = W\mathbf{x}, \tag{5c}$$

inserting this assumed form of the solution into the combined matrix form (5a) to eliminate $\mathbf{p}$, combining rows to eliminate $d\mathbf{x}/dt$, factoring out $\mathbf{x}$ to the right, and noting that this equation holds for all $\mathbf{x}$, it follows that the matrix $W(t) \geq 0$ itself obeys the differential Riccati equation (DRE)

$$\begin{aligned} -dW(t)/dt = {} & A^HW(t) + W(t)A \\ & - W(t)BR^{-1}B^HW(t) + Q\end{aligned} \tag{5d}$$

with terminal conditions

$$W(T) = Q_T. \tag{5e}$$

The solution $W(t)$ of the DRE (5d)-(5e) may be determined using any of a wide variety of well-known time marching methods, such as any of those in the Runge-Kutta family. By (4) and (5c), the control may then be determined according to the feedback rule

$$\mathbf{u} = K(t)\mathbf{x} \quad \text{where} \quad K(t) = -R^{-1}B^HW(t). \tag{5f}$$

Taking the limit as $T \to \infty$, assuming $\{A, B, Q, R\}$ are constant and thus the problem is linear time invariant (LTI), the DRE (5d) reduces to the continuous-time algebraic Riccati equation (CARE)

$$0 = A^HW + WA - WBR^{-1}B^HW + Q. \tag{6a}$$

The solution $W > 0$ of the CARE may be found [12] by taking an ordered Schur (or eigen[1]) decomposition of the Hamiltonian matrix[2] Z in (5a):

$$Z = VTV^{-1} \tag{6b}$$

[1] Numerically, the Schur decomposition is the method of choice for large-scale problems. In the analysis presented in §2, however, it is more convenient to consider the eigen decomposition.

[2] A Hamiltonian matrix of this form satisfies a symmetric root property: for every eigenvalue of Z in the LHP, λ, there is a corresponding eigenvalue of Z in the RHP, $-\lambda^*$, where $(\cdot)^*$ denotes the complex conjugate.

where

$$V = \begin{bmatrix} X & * \\ P & *\end{bmatrix} = \begin{bmatrix} | & | & & | & \\ \mathbf{v}^1 & \mathbf{v}^2 & \dots & \mathbf{v}^n & * \\ | & | & & | & \end{bmatrix}, \quad \mathbf{v}^i = \begin{bmatrix}\mathbf{x}^i \\ \mathbf{p}^i\end{bmatrix}, \tag{6c}$$

and the eigenvalues of Z on the main diagonal of the upper triangular (or diagonal) matrix T are enumerated such that the LHP eigenvalues appear first, followed by the RHP eigenvalues (that is, we assume that Z has no eigenvalues on the imaginary axis). Defining $\mathbf{y} = V^{-1}\mathbf{v}$, it follows from (5a) and (6b) that $d\mathbf{y}/dt = T\mathbf{y}$. The stable solutions of $\mathbf{y}$ are spanned by the first n columns of T (that is, they are nonzero only in the first n elements of $\mathbf{y}$). Since $\mathbf{v} = V\mathbf{y}$, it follows that the stable solutions of $\mathbf{v}$ are spanned by the first n columns of V. To achieve stability of $\mathbf{v}$ via the relation $\mathbf{p} = W\mathbf{x}$ for each of these directions, denoted $\mathbf{v}^i$ and decomposed as shown above, we must have $\mathbf{p}^i = W\mathbf{x}^i$ for $i = 1 \dots n$. Assembling these equations in matrix form, we have

$$\begin{bmatrix} | & | & & | \\ \mathbf{p}^1 & \mathbf{p}^2 & \dots & \mathbf{p}^n \\ | & | & & |\end{bmatrix} = W\begin{bmatrix} | & | & & | \\ \mathbf{x}^1 & \mathbf{x}^2 & \dots & \mathbf{x}^n \\ | & | & & |\end{bmatrix} \tag{6d}$$

$$\Rightarrow \quad P = WX \quad \Rightarrow \quad W = PX^{-1}. \tag{6e}$$

In order for this equation for W to be uniquely solvable, X must be nonsingular. Note also in this formulation that the evolution of $\mathbf{v}$ in (5a) is restricted by the relation $\mathbf{p} = W\mathbf{x}$ [i.e., noting (4) and (5f), by $\mathbf{u} = K\mathbf{x}$] to the space spanned by the stable eigenmodes of the matrix Z, and that the (LHP) eigenvalues corresponding to these stable eigenmodes of Z are exactly the eigenvalues of the closed-loop system matrix $(A + BK)$.

1.2 Chandrasekhar's method

The DRE of interest for $W = W_{n\times n}$ in the optimal control problem is given in (5d), and the corresponding expression for the feedback gain matrix $K = K_{m\times n}$ is given in (5f). If $n \gg m$, which is typical in high-dimensional systems, then solving a Riccati equation for the enormous $n \times n$ matrix W only to extract a transformed narrow "slice" of this matrix to determine the $m \times n$ matrix K is grossly inefficient. Chandrasekhar's method [11] addresses this inefficiency in a clever way, by simultaneously solving an evolution equation for a low-rank factor $F(t)$ of (dW/dt), and another evolution equation for $K(t)$. Towards this end, define

$$dW(t)/dt = F_1(t)\,F_1^H(t) - F_2(t)\,F_2^H(t) = F(t)\,H\,F^H(t)$$

where

$$F = \begin{pmatrix} F_1 & F_2\end{pmatrix}, \quad H = \begin{pmatrix} I & 0 \\ 0 & -I\end{pmatrix},$$

and the number of columns of the factors F_1 and F_2 are the number of positive and negative eigenmodes of (dW/dt), respectively, retained in the approximation. Differentiating (5d) with respect to time and inserting $dW/dt = FHF^H$, assuming $\{A, B, Q, R\}$ are LTI, it is easily verified that the following set of equations are equivalent to (5d), but much cheaper to compute if both K and F each has a relatively small number of columns:

$$dK(t)/dt = -R^{-1}B^H F(t)\, H\, F^H(t),$$
$$dF(t)/dt = -[A + B\, K(t)]^H F(t),$$

with terminal conditions

$$K(T) = -R^{-1}B^H Q_T,$$
$$F(T)\, H\, F^H(T) = dW(t)/dt\big|_{t=T},$$

where $dW/dt|_{t=T}$ is determined from the original DRE (5d) evaluated at $t = T$, and $F(T)$ is determined by its factorization. Chandrasekhar's method may be used to approximate the (time-accurate) solution of the DRE (5d), or simply marched to steady state to approximate the solution of the corresponding CARE (6a).

2 MCE: Minimum Control Energy stabilization

Selecting $Q = \epsilon I$, $Q_T = 0$, and taking any $R > 0$ in the derivation summarized in §1.1, and then taking ϵ small, puts the dominant weighting on the control effort term $(\mathbf{u}^H R\mathbf{u})$ in the cost function J in (1). The solution of the optimal control problem on the infinite horizon $(T \to \infty)$ in the limit that $\epsilon \to 0$ is referred to as the *minimum energy stabilizing control feedback*. Such feedback applies as little control effort as possible in order to make J finite; in other words, control applied in the $\epsilon \to 0$ limit leaves those modes of the system which are already stable alone, and swings the unstable eigenvalues of the system into the LHP in a way that uses the minimum amount of control effort. It is seen immediately from the form of Z in (5a) that the stable eigenvalues of Z in this case (with $Q \to 0$) are given by the union of the stable eigenvalues of A and the stable eigenvalues of $-A^H$ (that is, the reflection of the unstable eigenvalues of A into the LHP across the imaginary axis); by the last sentence of §1.1, this is where the eigenvalues of $(A + BK)$ in the optimal control solution are as well. This result is classical.

Since we know where the eigenvalues of the closed-loop system matrix $(A+BK)$ are in this case, the requisite feedback gain matrix K in this problem may be computed by the process of *pole assignment*. Applying this process to the equation governing the dynamics of the unstable modes of the system in modal form and transforming appropriately, this leads to a simple expression for K, as shown below and discussed in [14].

2.1 The pole assignment problem

Consider now the Hamiltonian matrix Z in (5a) and its eigen decomposition in (6b). Defining a diagonal matrix Λ_c with the n desired (stable) eigenvalues λ_c of the closed-loop system on the main diagonal, and the corresponding eigenvectors of Z in the columns of V_c (which may be partitioned appropriately), the stable components of the eigen decomposition of Z satisfy

$$\underbrace{\begin{bmatrix} A & -BR^{-1}B^H \\ -Q & -A^H \end{bmatrix}}_{Z} V_c = V_c \Lambda_c \quad \text{where} \quad V_c = \begin{bmatrix} X \\ P \end{bmatrix}. \quad (7)$$

In a typical pole assignment problem, the closed-loop eigenvalues λ_c are prescribed in advance, then the control feedback matrix K [equivalently, the off-diagonal blocks of the matrix on the LHS of (7) in the optimal control problem] is selected in order to put these eigenvalues in the desired locations.

In the present pole assignment problem, however, we happen to know both the closed-loop eigenvalues λ_c *and* the off-diagonal blocks of Z in (7); all that remains is for us to compute the corresponding eigenvector matrix V_c. Exactly as in (5f) and (6e), once these eigenvectors are calculated, the desired feedback rule is given by

$$\mathbf{u} = K\mathbf{x} \quad \text{with} \quad K = -R^{-1}B^H W \quad (8a)$$
$$\text{where} \quad W = PX^{-1}. \quad (8b)$$

Multiplying out (7), it follows immediately that

$$AX - BR^{-1}B^H P = X\Lambda_c, \quad (9a)$$
$$-QX - A^H P = P\Lambda_c. \quad (9b)$$

Solving (9b) for X and substituting into (9a) gives

$$AQ^{-1}(A^H P + P\Lambda_c) + BR^{-1}B^H P = Q^{-1}(A^H P + P\Lambda_c)\Lambda_c \quad (10a)$$

and

$$QX = -(A^H P + P\Lambda_c). \quad (10b)$$

Note (10a) is linear in the unknown matrix P. Once a nonsingular P is obtained from this equation, calculation of X is trivial using (10b) or, equivalently, (9a).

2.2 Simplification of (8)-(10) in modal coordinates

It is straightforward to transform the original system (2a) into a modal representation and then to truncate appropriately in order to develop a model of just the unstable system dynamics. Performing an ordered eigen decomposition $A = S\Lambda S^{-1}$, in which the unstable eigenvalues of A appear on the main diagonal of Λ first, followed by the stable eigenvalues of A,

$$\Lambda = \begin{bmatrix} \Lambda_u & 0 \\ 0 & \Lambda_s \end{bmatrix},$$

and multiplying (2a) from the left by S^{-1}, we have

$$d\boldsymbol{\chi}/dt = \Lambda\boldsymbol{\chi} + \bar{B}\mathbf{u} \quad \text{where} \quad \boldsymbol{\chi} = S^{-1}\mathbf{x}, \;\; \bar{B} = S^{-1}B. \tag{11}$$

Note that Λ is diagonal. Denoting the inverse of the eigenvector matrix as[3] $Y^H = S^{-1}$, the portion of (11) governing the unstable dynamics of the system may easily be written as

$$d\boldsymbol{\chi}^u/dt = \Lambda_u \boldsymbol{\chi}^u + \bar{B}_u\mathbf{u} \tag{12}$$

where

$$Y = \begin{bmatrix} Y_u & Y_s \end{bmatrix}, \quad \bar{B} = \begin{bmatrix} \bar{B}_u \\ \bar{B}_s \end{bmatrix}, \quad \bar{B}_u = Y_u^H B, \quad \boldsymbol{\chi}^u = Y_u^H\mathbf{x}.$$

The pole assignment process in the minimal-energy stabilizing feedback control problem, as derived in §2.1, can be simplified greatly when applied to the equation for the unstable dynamics of the original system in modal form, as given in (12). Assuming A has p unstable eigenvalues, taking $A = \Lambda_u$, $B = \bar{B}_u$, $Q = \epsilon I$, $R > 0$, and[4] $\Lambda_c = -\Lambda_u^H$ in (10a), partitioning P into its respective columns, $P = \begin{bmatrix}\mathbf{p}^1 \; \mathbf{p}^2 \; \ldots \; \mathbf{p}^p\end{bmatrix}$, and applying the above relationships, it follows after some simplifications[5] that (10a) may be written in the simple form

$$[\epsilon\bar{B}_u R^{-1}\bar{B}_u^H + D]\mathbf{p}^k \triangleq M_k\mathbf{p}^k = 0 \tag{13}$$

for $k = 1, \ldots, p$, where $D = \text{diag}(d_1^{(k)}, \ldots, d_p^{(k)})$ with

$$d_i^{(k)} = \begin{cases} (\lambda_i + \lambda_k^*)(\lambda_i^* - \lambda_k^*) & \text{for } i \neq k \\ 0 & \text{for } i = k, \end{cases} \tag{14}$$

where, again, $(\cdot)^*$ denotes the complex conjugate. Thus, the vectors $\mathbf{p}^k$ lie in the nullspace of M_k, and may be found by the process of Gaussian elimination, manipulating M_k to row-echelon form. In the limit $\epsilon \to 0$, M_k approaches a diagonal matrix with a zero in the k'th diagonal element, and thus[6] $P \to I$. To avoid taking the difference of two quantities which are almost equal in the computation of X, we return to (9a), which, in the $\epsilon \to 0$ limit, may be written in the form

$$\Lambda_u X + X\Lambda_u^H = \bar{B}_u R^{-1}\bar{B}_u^H \triangleq C \tag{15a}$$

$$\Rightarrow \quad x_{ij} = c_{ij}/(\lambda_i + \lambda_j^*). \tag{15b}$$

With $P = I$, it follows from (8b) that $W = X^{-1}$, and thus, by (8a), the minimal-energy feedback control that stabilizes (12) in the $\epsilon \to 0$ limit is given by $\mathbf{u} = \bar{K}\boldsymbol{\chi}^u$ where $\bar{K} = -R^{-1}\bar{B}_u^H X^{-1}$. Writing this feedback in terms of the original state variable $\mathbf{x}$, we have $\mathbf{u} = K\mathbf{x}$ where $K = \bar{K}Y_u^H$.

The solution for the minimum control energy (MCE) controller derived above is now summarized:

Theorem 1. *Consider a stabilizable system $d\mathbf{x}/dt = A\mathbf{x} + B\mathbf{u}$ where A has no pure imaginary eigenvalues. Determine the unstable eigenvalues and corresponding left eigenvectors of A such that $Y_u^H A = \Lambda_u Y_u^H$ (equivalently, determine the unstable eigenvalues and corresponding right eigenvectors of A^H such that $A^H Y_u = Y_u \Lambda_u^H$). Define $\bar{B}_u = Y_u^H B$ and $C = \bar{B}_u R^{-1}\bar{B}_u^H$, and compute a matrix X with elements $x_{ij} = c_{ij}/(\lambda_i + \lambda_j^*)$. The minimal-energy stabilizing feedback controller is then given by $\mathbf{u} = K\mathbf{x}$, where $K = -R^{-1}\bar{B}_u^H X^{-1} Y_u^H$.*

Note that any of a number of existing subspace iteration methods (see, e.g., §4.1) may be used to determine the unstable eigenvalues (that is, those eigenvalues on the far right edge of the spectrum of eigenvalues) and corresponding left eigenvectors of A; the implicitly-restarted Arnoldi method [15] is a popular choice.

2.3 Alternative derivation of the MCE method

The derivation of the MCE method given in §2.1 - 2.2 is based on pole assignment in the Riccati equation in the minimum control energy case. An alternative derivation of this method based on analysis of the corresponding state and adjoint vectors is now presented; this alternative derivation foreshadows the methods developed in §3 and §4, which are similarly based on careful analysis of the state and adjoint components of the Hamiltonian.

[3] The columns of Y are referred to as the *left* or *adjoint* eigenvectors of A.

[4] We take $\Lambda_c = -\Lambda_u^H$ following the first paragraph of §2, noting that *all* eigenvalues in Λ_u are unstable.

[5] If Λ is diagonal, the product ΛV corresponds to scaling the i'th *row* of V by λ_i for all i, whereas the product $V\Lambda$ corresponds to scaling the i'th *column* of V by λ_i for all i.

[6] If all unstable eigenvalues of A are distinct, then $d_i^{(k)} \neq 0$ for $i \neq k$; P necessarily becomes diagonal in this case in the $\epsilon \to 0$ limit, and its columns may be normalized such that $P \to I$. If some of the unstable eigenvalues of A are repeated, then there are other solutions as well. However, $P \to I$ is a valid solution in either case in the $\epsilon \to 0$ limit.

Taking $Q = \epsilon I$ in the limit that $\epsilon \to 0$, the Hamiltonian system (5a) may be written

$$\frac{d}{dt}\underbrace{\begin{bmatrix}\mathbf{x}\\ \mathbf{p}\end{bmatrix}}_{\mathbf{v}} = \underbrace{\begin{bmatrix}A & -BR^{-1}B^H\\ 0 & -A^H\end{bmatrix}}_{Z}\underbrace{\begin{bmatrix}\mathbf{x}\\ \mathbf{p}\end{bmatrix}}_{\mathbf{v}}, \tag{16a}$$

which may equivalently be written

$$d\mathbf{x}/dt = A\mathbf{x} + B\mathbf{u}, \quad \mathbf{u} = -R^{-1}B^H\mathbf{p}, \tag{16b}$$

$$d\mathbf{p}/dt = -A^H\mathbf{p}. \tag{16c}$$

Due to the block triangular structure of Z, the eigenvalues of Z in this limit are simply the union of the eigenvalues of A (that is, λ_k for $k = 1, \dots, n$) and the eigenvalues of $-A^H$ (that is, $-\lambda_k^*$ for $k = 1, \dots, n$). Let p be the number of unstable eigenvalues of A, denote by $\mathbf{s}^k$ the eigenvectors of A [that is, $A\mathbf{s}^k = \lambda_k\mathbf{s}^k$], and denote by $\mathbf{y}^k$ the eigenvectors of A^H [that is, $A^H\mathbf{y}^k = \lambda_k^*\mathbf{y}^k$]; note that the $\mathbf{y}^k$ are also known as the left eigenvectors of A [that is, $(\mathbf{y}^k)^H A = \lambda_k(\mathbf{y}^k)^H$], and that the $\mathbf{y}^i$ and $\mathbf{s}^k$ vectors are orthogonal for $i \neq k$ and may be normalized such that

$$(\mathbf{y}^i)^H\mathbf{s}^k = \delta_{ik}. \tag{17}$$

Assuming that A has no pure imaginary eigenvalues, the n eigenvectors $\mathbf{v}^k$ of Z corresponding to the stable eigenvalues of Z are given, for $k = 1, \dots, n$, by

$$\mathbf{v}^k = \begin{bmatrix}\mathbf{x}^k\\ \mathbf{p}^k\end{bmatrix} = \begin{cases}\begin{bmatrix}\mathbf{s}^k\\ 0\end{bmatrix} & \text{if } \Re(\lambda_k) < 0,\\[2mm] \begin{bmatrix}\mathbf{f}^k\\ \mathbf{y}^k\end{bmatrix} & \text{if } \Re(\lambda_k) > 0,\end{cases} \tag{18}$$

where, by the first block row of (16a),

$$\mathbf{f}^k = (A + \lambda_k^* I)^{-1}BR^{-1}B^H\mathbf{y}^k. \tag{19}$$

Let $I_s = \{\text{all } k \mid \Re(\lambda_k) < 0\}$ and $I_u = \{\text{all } k \mid \Re(\lambda_k) > 0\}$. Expanding the current state $\mathbf{x}$ in terms of the $\mathbf{x}^k$ components of the stable eigenmodes $\mathbf{v}^k$ given in (18), and expanding the corresponding adjoint $\mathbf{p}$ in a compatible manner, we have

$$\mathbf{x} = \sum_{k\,\in\, I_s}\mathbf{s}^k c_k + \sum_{k\,\in\, I_u}\mathbf{f}^k d_k, \tag{20a}$$

$$\mathbf{p} = \sum_{k\,\in\, I_u}\mathbf{y}^k d_k, \tag{20b}$$

for the as-yet undetermined coefficients c_k for $k \in I_s$, and d_k for $k \in I_u$, which we assemble into the vectors $\mathbf{c}$ and $\mathbf{d}$, respectively. Note in particular [in (20b)] that the adjoint field $\mathbf{p}$ upon which the control $\mathbf{u}$ is based [in (16b)] is itself based solely upon the coefficients d_k. To compute these coefficients, we simply premultiply (20a) by $(\mathbf{y}^i)^H$ for all $i \in I_u$ and apply the orthogonality (17):

$$(\mathbf{y}^i)^H\left\{\mathbf{x} = \sum_{k\,\in\, I_s}\mathbf{s}^k c_k + \sum_{k\,\in\, I_u}\mathbf{f}^k d_k\right\}$$

$$\Rightarrow \quad (\mathbf{y}^i)^H\mathbf{x} = \sum_{k\,\in\, I_u}(\mathbf{y}^i)^H\mathbf{f}^k d_k. \tag{21}$$

Premultiplying (19) by $[(\mathbf{y}^i)^H(A + \lambda_k^* I)]$ results in

$$(\mathbf{y}^i)^H(A + \lambda_k^* I)\,\mathbf{f}^k = (\mathbf{y}^i)^H BR^{-1}B^H\mathbf{y}^k,$$

and thus, since $(\mathbf{y}^i)^H A = \lambda_i(\mathbf{y}^i)^H$, it follows that

$$(\mathbf{y}^i)^H\mathbf{f}^k = \frac{(\mathbf{y}^i)^H BR^{-1}B^H\mathbf{y}^k}{\lambda_i + \lambda_k^*} \triangleq x_{ik}. \tag{22a}$$

Collecting the vectors $\mathbf{y}^i$ for $i \in I_u$ together as the columns of a matrix Y_u and noting the definition of the elements of X in (22a), we may write (20b) and (21) in matrix form as

$$\mathbf{p} = Y_u\mathbf{d} \quad \text{and} \quad Y_u^H\mathbf{x} = X\mathbf{d} \quad \Rightarrow \quad \mathbf{d} = X^{-1}Y_u^H\mathbf{x}$$

and thus, by (16b),

$$\mathbf{u} = K\mathbf{x} \quad \text{where} \quad K = -R^{-1}B^H Y_u X^{-1} Y_u^H, \tag{22b}$$

as summarized in Theorem 1.

3 ADA: the Adjoint of the Direct-Adjoint

As described in the first paragraph (and verified in the second paragraph) of §1.1, for any given $\mathbf{x}_0$, adjoint optimization of $\mathbf{u}$ for minimization of $J(\mathbf{x}(\mathbf{u}), \mathbf{u})$ in (1) (taking $Q_T = 0$) proceeds as indicated in Algorithm 1; when this algorithm converges, $DJ(\mathbf{x}(\mathbf{u}), \mathbf{u})/D\mathbf{u} = 0$, and thus the following optimality condition holds:

$$\mathbf{u} = -R^{-1}B^H\mathbf{p} \quad \text{on} \quad t \in [0, T]. \tag{23}$$

The input to this problem is $\mathbf{x}_0$, and we will focus for now on the output $\mathbf{u}$ at time $t = 0$, which we denote here $\mathbf{u}_0$. If $\mathbf{x} = \mathbf{x}_{n\times 1}$ and $\mathbf{u} = \mathbf{u}_{m\times 1}$ and we solve this problem n times for n linearly independent values of $\mathbf{x}_0$ (e.g., the n columns of $I_{n\times n}$), then we may write

$$\left[\mathbf{u}_0^1\ \mathbf{u}_0^2 \dots \mathbf{u}_0^n\right] = K_0\left[\mathbf{x}_0^1\ \mathbf{x}_0^2 \dots \mathbf{x}_0^n\right] \tag{24}$$

and solve for K_0, thus determining the feedback gain matrix K at time $t = 0$,

$$\mathbf{u}(0) = K_0\mathbf{x}(0). \tag{25}$$

Note also that (25) taken together with (23), evaluated at $t = 0$, may be written

$$\mathbf{u}(0) = [K_0^H]^H\mathbf{x}(0) = [-BR^{-1}]^H\mathbf{p}(0). \tag{26}$$

Algorithm 1 Direct-adjoint problem for optimizing $\mathbf{u}$, framing (1) and (2a)-(2c) as a numerical algorithm (taking $Q_T = 0$).

```
Initialize $i = 0$, $\mathbf{u}(t) = 0$ on $t \in [0, T]$, and the tolerance $\epsilon$
loop
    March $d\mathbf{x}/dt = A\mathbf{x} + B\mathbf{u}$ with $\mathbf{x}(0) = \mathbf{x}_0$ for $t = 0 \to T$
    Compute $J_i = J(\mathbf{x}(\mathbf{u}), \mathbf{u})$ [see (1)]
    if $i > 0$ and $(J_i - J_{i-1})/J_i < \epsilon$ then break end if
    March $d\mathbf{p}/dt = -A^H\mathbf{p} - Q\mathbf{x}$ with $\mathbf{p}(T) = 0$ for $t = T \to 0$
    Compute gradient $DJ(\mathbf{x}(\mathbf{u}), \mathbf{u})/D\mathbf{u} = B^H\mathbf{p} + R\mathbf{u}$
    Update $\mathbf{u}$ via gradient-based algorithm (CG, BFGS, ...)
end loop
```

The approach described above requires n optimizations to set up (24), which may be solved to compute the $m \times n$ matrix K_0. If $n \gg m$, it is much more efficient to consider the adjoint of this problem, thus leading to an algorithm requiring only m optimizations to compute K_0, which can be achieved as described below.

To frame the adjoint of the problem described above in a manner analogous to the standard adjoint-based optimization framework reviewed in (3a)-(3g), we first introduce a bit of notation. Define

$$\mathbf{y} = \begin{bmatrix} \mathbf{p} \\ \mathbf{x} \end{bmatrix} \quad \text{and} \quad \mathcal{L} = \begin{bmatrix} BR^{-1}B^H & d/dt - A \\ -d/dt - A^H & -Q \end{bmatrix} \tag{27a}$$

where $Q \geq 0$ and $R > 0$, and note that the converged solution of the "forward TPBVP" (2a)-(2b) with (4) and $Q_T = 0$, which may be calculated using Algorithm 1, may now equivalently be written

$$\mathcal{L}\mathbf{y} = 0 \quad \text{with} \quad \begin{cases} \mathbf{x}(0) = \mathbf{x}_0, \\ \mathbf{p}(T) = 0. \end{cases} \tag{27b}$$

[Note that, in (27a), we have arranged the variables and equations appropriately to include the d/dt operator in the *off-diagonal* blocks of the linear operator $\mathcal{L}$; this slightly nonstandard approach simplifies and symmetrizes the subsequent analysis.] Defining as before the inner product $\langle \mathbf{a}, \mathbf{b} \rangle = \int_0^T \mathbf{a}^H \mathbf{b}\, dt$, we may express the adjoint identity

$$\langle \tilde{\mathbf{y}}, \mathcal{L}\mathbf{y} \rangle = \langle \mathcal{L}^* \tilde{\mathbf{y}}, \mathbf{y} \rangle + b \quad \text{where} \quad \tilde{\mathbf{y}} = \begin{bmatrix} \tilde{\mathbf{p}} \\ \tilde{\mathbf{x}} \end{bmatrix}. \tag{27c}$$

Using integration by parts, it follows immediately that $\mathcal{L}^* = \mathcal{L}$ (that is, the state/adjoint TPBVP considered as a whole is itself *self-adjoint*), and that

$$b = (\tilde{\mathbf{p}}^H \mathbf{x} - \tilde{\mathbf{x}}^H \mathbf{p})_{t=T} - (\tilde{\mathbf{p}}^H \mathbf{x} - \tilde{\mathbf{x}}^H \mathbf{p})_{t=0}. \tag{27d}$$

[Note the permutation (that is, $\tilde{\mathbf{p}}^H$ multiplies $\mathbf{x}$, whereas $\tilde{\mathbf{x}}^H$ multiplies $\mathbf{p}$), which arises due to the off-diagonal location of the d/dt terms in $\mathcal{L}$.] We now use $\mathcal{L}^*$ to define an appropriate adjoint equation

$$\mathcal{L}^* \tilde{\mathbf{y}} = 0 \quad \text{with} \quad \begin{cases} \tilde{\mathbf{x}}(0) = \tilde{\mathbf{x}}_0, \\ \tilde{\mathbf{p}}(T) = 0, \end{cases} \tag{27e}$$

which may equivalently be written

$$d\tilde{\mathbf{x}}/dt = A\tilde{\mathbf{x}} + B\underbrace{(-R^{-1}B^H\tilde{\mathbf{p}})}_{\triangleq \tilde{\mathbf{u}}} \quad \text{with} \quad \tilde{\mathbf{x}}(0) = \tilde{\mathbf{x}}_0, \tag{28a}$$

$$-d\tilde{\mathbf{p}}/dt = A^H\tilde{\mathbf{p}} + Q\tilde{\mathbf{x}} \quad \text{with} \quad \tilde{\mathbf{p}}(T) = 0. \tag{28b}$$

By (27a)-(27b), this TPBVP is *exactly* the same as that given in (2a), (2b), and (4), it is just written in different variables, and has a different interpretation given to the "input" $\tilde{\mathbf{x}}(0)$ and the "output" $\tilde{\mathbf{p}}(0)$. Thus, this "adjoint TPBVP" may also be solved using Algorithm 1.

The key to relate the solution of the "adjoint TPBVP" (27e) to the "forward TPBVP" (27b) is the adjoint identity (27c), which reduces upon substitution of (27d), (27b), and (27e) to

$$[\tilde{\mathbf{p}}(0)]^H \mathbf{x}(0) = [\tilde{\mathbf{x}}(0)]^H \mathbf{p}(0). \tag{29}$$

Comparing (29) and (26), it is seen that, setting $\tilde{\mathbf{x}}(0)$ to the first column of $[-BR^{-1}]$ and solving the adjoint TPBVP (27e) via Algorithm 1, the resulting value of $\tilde{\mathbf{p}}(0)$ is just the first column of K_0^H, etc. Thus, after solving the adjoint TPBVP (27e) via Algorithm 1 just m times (not n times!), the entire K_0 is constructed directly. Further, for an LTI system and T sufficiently large, K_0 approximates the LTI feedback gain K.

3.1 Interpretation of the ADA method

The optimality condition $DJ(\mathbf{x}(\mathbf{u}), \mathbf{u})/D\mathbf{u} = 0$ relating the state $\mathbf{x}$ to the control $\mathbf{u}$ is linear, as it is given by the first derivative of a quadratic cost function $J(\mathbf{x}(\mathbf{u}), \mathbf{u})$ with respect to $\mathbf{u}$, where $\mathbf{x}$ is a linear function of $\mathbf{u}$ via the state equation. The linearity of this problem is sometimes obscured by the fact that this relationship is usually solved by considering a quadratic matrix equation (the Riccati equation); the purpose for introducing this quadratic matrix equation is simply to convert the TPBVP for the vector $\mathbf{x}$ and a supplemental vector $\mathbf{p}$ at the core of this linear problem into an initial value problem for a matrix W that relates these two vectors.

For low-dimensional systems, this quadratic initial value problem for the matrix W is easy to solve [in the infinite-horizon LTI case, leveraging eigen- or Schur-based analysis, as reviewed in §1.1]. For high-dimensional systems, however, computation of W is intractable, and in certain situations it is useful to reconsider to the optimality condition in its original linear form.

Since the optimality condition is linear, it may be considered in one of two directions, a "forward" analysis (itself based on iterative solution of a direct/adjoint formulation), which as shown in §1.1 comes up naturally when examining the control problem, and the "adjoint" of this analysis. If the state $\mathbf{x}$ and the control $\mathbf{u}$ are roughly the same dimension, both of these directions have similar computational expense. However, if the dimension of the state $\mathbf{x}$ is significantly larger than that of the control $\mathbf{u}$, the "adjoint" analysis of this problem is significantly more efficient computationally.

Finally, since the state/adjoint pair at the core of the "forward" analysis is itself a self-adjoint system, exactly the same numerical machinery may be used for the "forward" and "adjoint" analyses, it is only the inputs and outputs to these analyses that change.

4 OSSI: Oppositely-Shifted Subspace Iteration

"Subspace iteration" refers to the core framework of a class of iterative eigenvalue solvers designed to extract $m \ll n$ eigenvalues, and the corresponding eigenvectors or Schur vectors, from an $n \times n$ matrix A when $n \gg 1$. Two modern extensions of such solvers, dubbed Arnoldi (for general matrices) and Lanczos (for symmetric matrices), are today commonly applied to sparse systems with $n \geq O(10^6)$. There are a wide variety of well-established techniques available for accelerating the convergence of such algorithms, including deflating and implicit restarting, which for brevity will not be expounded upon here; some reviews of this fascinating and now fairly mature subject include [22,17,20,8,15].

In §4.1, we describe two "prototype" subspace iteration algorithms, including an explicit form and an implicit form, which demonstrate the essence of how such algorithms may be used to extract a basis of the eigenvectors and Schur vectors corresponding to the m most unstable eigenvalues of a matrix A. It is important to note that the explicit form considered needs access solely to a subroutine which computes the matrix/vector product $A\mathbf{v}$; it does not need access to A^{-1}, nor even to A itself. This is useful in many complex applications with $n \geq O(10^6)$, such as those arising in the field of flow control, where a subroutine which effectively computes the matrix/vector product $A\mathbf{v}$ is often all that is available. An implicit form is also considered, which is built around a subroutine designed to efficiently solve $(1+hA)\mathbf{x} = \mathbf{v}$ for $\mathbf{x}$, which is also sometimes available and numerically tractable (and, when it is, can be leveraged to significantly speed convergence of the subspace iteration algorithm when A is *stiff*, meaning that it has a large range of eigenvalues reaching far out into the LHP, as is common in the spatial discretization of PDE systems [9]). Significantly, many such implicit solvers are only approximate, such as those based on incomplete Cholesky factorization (for symmetric A), or those on based on iterative solvers (such as the multigrid algorithm) when such solvers are not driven fully to convergence[7]. Note also that the subroutine which computes the matrix/vector product $A\mathbf{v}$ in the explicit case, or the subroutine which solves $(1+hA)\mathbf{x} = \mathbf{v}$ for $\mathbf{x}$ in the implicit case, is called only m times per iteration[8] in the core algorithms presented; this is important because these subroutines are typically computationally expensive.

In §4.2, we demonstrate how a simple modification, dubbed "opposite shifting", of these prototype subspace iteration algorithms may be used to target the *central* eigenvalues (that is, those near the imaginary axis) of a matrix Z of Hamiltonian structure. The resulting oppositely-shifted subspace iteration (OSSI) algorithms are again developed in prototype explicit and implicit forms, and are quite similar to the explicit and implicit forms of the standard subspace iteration algorithms upon which they are based. The OSSI algorithms proposed again depend only on the matrix/vector product $A\mathbf{v}$ in the explicit case, or on the (possibly, approximate) solution of $(1 + hA)\mathbf{x} = \mathbf{v}$ for $\mathbf{x}$ in the implicit case (that is, they do not require direct access to, or storage of, A or A^{-1}). Once the simple "opposite shifting" modification is made, the wide variety of techniques available to accelerate the convergence of such algorithms are again directly applicable. [For brevity, we will not focus on such acceleration techniques here.]

Finally, in §5, we discuss the application of these prototype OSSI algorithms to the approximate solution of large eigenvalue problems arising in optimal control formulations while bypassing the preparatory (and, sometimes, problematical) step of open-loop model reduction, and briefly compare these algorithms to the other strategies for the control of large-scale systems discussed previously in this article.

4.1 Prototype subspace iteration algorithms

To understand the basic idea of standard subspace iteration methods, consider first the simple ODE

$$d\mathbf{v}/dt = A\mathbf{v}, \tag{30}$$

[7] In future work, it would be valuable to consider the myriad of subtle issues that arise when coupling the implicit OSSI algorithm developed in §4.2 with approximate inverses such as those arising in the multigrid setting.

[8] That is, once per column of V being computed [see (31b) and (39b)].

and order the eigenvalues λ_i and corresponding eigenvectors $\mathbf{s}^i$ of A by the real parts of λ_i [that is, λ_1 is the eigenvalue of A with the greatest real part, and λ_n is the eigenvalue of A with the least real part; note that high-dimensional ODE systems arising from the spatial discretization of diffusive PDE systems typically have just a few unstable eigenvalues]. The utility of considering this ODE is that, as it evolves, it preferentially amplifies the component of $\mathbf{v}$ in the direction $\mathbf{s}^1$ as compared with the other components of $\mathbf{v}$.

4.1.1 Subspace iteration via EE discretization of (30)

Based on the above discussion, we are motivated to march (30) with a simple explicit Euler (EE) numerical discretization,

$$\mathbf{v}^{k+1} = (I + h\,A)\mathbf{v}^k = \mathbf{v}^k + h\,A\,\mathbf{v}^k, \tag{31a}$$

while using as large a timestep h as possible to accelerate the relative growth of the component of $\mathbf{v}^k$ in the direction $\mathbf{s}^1$ from one timestep to the next. Since the EE method is conditionally stable[9], it is actually the *most* stable eigenvalue, λ_n, that typically limits how large a timestep h can be taken in this march while not encountering the "numerical instability" caused by $|1 + h\lambda_n|$ approaching and exceeding 1.

To find a basis for the first $m > 1$ eigenvectors and Schur vectors, rather than propagating a single direction $\mathbf{v}$ as in (31a), we instead simply propagate a set of directions assembled as the columns of a matrix V,

$$V \leftarrow V + h\,A\,V. \tag{31b}$$

At the end of each iteration in the subspace iteration algorithm, we orthogonalize and normalize the columns of V via the $\underline{Q}\,\underline{R}$ decomposition using a Modified Gram-Schmidt method performed in place[10] [8], thus assuring that the m directions so generated are orthogonal:

$$V = \underline{Q}\,\underline{R}, \quad V \leftarrow \underline{Q}, \quad \Sigma \leftarrow \underline{R}\,\Sigma\,\underline{R}^{-1}, \tag{32}$$

where the columns of the updated V are orthogonalized and both $\underline{R}$ and the updated Σ are upper triangular.

An alternative motivation for (31b) is a bit more algebraic: for sufficiently small h, the largest (in modulus) eigenvalues of the shifted matrix $(I + h\,A)$ are $(1 + h\lambda_1)$ through $(1 + h\lambda_m)$, and the corresponding eigenvectors are, again, $\mathbf{s}^1$ through $\mathbf{s}^m$. Thus, the difference equation (31b), as it evolves, preferentially amplifies the components of V in the directions $\mathbf{s}^1$ through $\mathbf{s}^m$ as compared with the other components of V. For sufficiently small h, the next largest eigenvalue of $(I+h\,A)$ is $(1+h\lambda_{m+1})$, and the rates at which the components of V in the (resolved) directions $\mathbf{s}^1$ through $\mathbf{s}^m$ emerge over the components in the (unresolved) direction $\mathbf{s}^{m+1}$ are

$$\left|\frac{1 + h\lambda_1}{1 + h\lambda_{m+1}}\right| \quad \text{through} \quad \left|\frac{1 + h\lambda_m}{1 + h\lambda_{m+1}}\right|, \tag{33a}$$

whereas the rate at which the components of V in the direction $\mathbf{s}^{m+1}$ are amplified over the components in the direction $\mathbf{s}^n$ is

$$\left|\frac{1 + h\lambda_{m+1}}{1 + h\lambda_n}\right|. \tag{33b}$$

We desire to select h as large as possible, in order to make the ratios in (33a) large, without making h so large that the ratio in (33b) exceeds 1. As the λ_i are typically unknown, optimizing the value of h for a given problem in this manner often takes a modest amount of trial and error[11].

Another key step of a subspace iteration algorithm is to extract the eigenvalues corresponding to the emerging eigenvector subspace. There are a few different options for this. In the case that $m = 1$ and A is symmetric, we may simply use the Rayleigh quotient (see [8], p. 408): normalizing $\mathbf{v}$ at the end of each iteration such that $\mathbf{v}^H\mathbf{v} = 1$ and denoting our estimate of the eigenvalue corresponding to emerging eigenvector in $\mathbf{v}$ as σ, this may be written

$$\sigma = \mathbf{v}^H\,A\,\mathbf{v}. \tag{34a}$$

As $\mathbf{v}$ converges towards a (normalized) eigenvector, the value of σ computed in this manner converges quickly towards the corresponding eigenvalue.

In order to approximate the eigenvalues of the emerging subspace in the more interesting case that A is nonsymmetric and/or $m > 1$, we may follow a similar approach: orthogonalizing V at the end of each iteration such that $V^H V = I$, we impose

$$\Sigma = V^H\,A\,V. \tag{34b}$$

[9] Recall that the EE method is stable when all eigenvalues λ, scaled by h, are contained in a unit disk centered at -1 in the complex plane of $h\lambda$.

[10] That is, in a manner immediately replacing V with $\underline{Q}$.

[11] Various approaches are available to "stretch" the eigenvalue spectrum of A in order to mitigate this timestep restriction. For example, one may replace the matrix A to which the iteration is applied with an appropriately-designed low-order polynomial (of order p) in A; such a polynomial has the same eigenvectors as A. This approach helps to increase the gap between λ_m and λ_{m+1} [see (33a)] while decreasing the interval between λ_{m+1} and λ_n [see (33b)], both of which facilitate faster convergence. However, this approach also increases the number of function evaluations $A\mathbf{v}$ that must be calculated per iteration by a factor of p, and is thus generally not worthwhile unless the range of the eigenvalue spectrum of A is, a priori, known fairly accurately.

To motivate this form, consider what happens when we upper triangularize the $(m \times m)$ matrix Σ at each step of the subspace iteration algorithm via the computation of an ordered Schur decomposition ([8], p. 313)

$$\Sigma = \overline{U}\,\overline{T}\,\overline{U}^H, \tag{35a}$$

where $\overline{T} = \overline{T}_{m\times m}$ is upper triangular, with its diagonal elements arranged in order of decreasing real part, and $\overline{U} = \overline{U}_{m\times m}$ is unitary. [In the case that A is real, a real Schur decomposition ([8], p. 341) should be used instead at this step; rather than generating a complex upper-triangular matrix T, the real Schur decomposition generates a real matrix $\overline{T}$ which is block upper triangular, with 1×1 blocks (corresponding to the real eigenvalues) and 2×2 blocks (corresponding to the complex-conjugate eigenvalue pairs) on the main diagonal.] Taking (34b) together with (35a) then gives

$$\overline{U}\,\overline{T}\,\overline{U}^H = V^H A\,V \quad \Rightarrow \quad \overline{T} = (V\overline{U})^H A\,(V\overline{U}). \tag{35b}$$

Note that updating

$$V \leftarrow (V\overline{U}) \quad \text{and} \quad \Sigma \leftarrow \overline{T} \tag{35c}$$

changes neither the subspace spanned by the columns of V, nor the fact that V is unitary, nor the eigenvalues of Σ, which in this triangularized form appear on the main diagonal (or, in the real Schur case, may be derived from the blocks on the main diagonal). Further, as V converges towards the first m Schur vectors of A via this process, Σ converges towards the $m \times m$ leading principal submatrix of T in the full Schur decomposition $A = UTU^H$.

For small n, overall convergence is not significantly affected if the Schur decomposition of Σ is deferred until after the end of the main loop; in this case, Σ is left in a full form as the iteration proceeds, and V converges toward an essentially arbitrary basis of the first m Schur vectors.

Note in (33a) that leading Schur vectors tend to converge faster than do subsequent Schur vectors. It is generally wise to include the Schur decomposition step (35a)-(35c), which triangularizes Σ and "unscrambles" the emerging Schur vectors in V, within the main iteration loop of the subspace iteration algorithm for large problems (though perhaps not at every single iteration step), as this step helps to separate the convergence of each of the Schur vectors being computed. This separation is useful for two reasons. The first is that it is sometimes necessary to terminate the subspace iteration algorithm before the last few Schur vectors being computed fully converge. The second is that, as the first few Schur vectors begin to converge, the problem being worked on may be successively *deflated* (that is, the converged Schur vectors may be removed from V), as these converged vectors do not need to be worked on further by the subspace iteration algorithm as it proceeds[12].

As a significant refinement, we may apply a small shift of $-(hV\Sigma)$ to (31b), thus marching

$$V \leftarrow V + h\,(A\,V - V\Sigma). \tag{36}$$

This shifted form has the same essential effect as marching (31b) (that is, preferential focusing of the columns of V in the direction of the leading Schur vectors of A), with the benefit of ensuring that the update to V itself approaches zero as V approaches a basis of a set of Schur vectors, and thus $(A\,V - V\Sigma)$ approaches zero. This is clearly seen in the $m = 1$ case, where this shift makes the amplification of $\mathbf{s}^1$ at each iteration unity, and (for sufficiently small h) the amplification of all other directions smaller. In the $m > 1$ case, the effect of this small shift $(hV\Sigma)$ can be thought of in terms its effect on the convergence of the individual columns of V: the shift in the first column makes the amplification of $\mathbf{s}^1$ at each iteration unity as in the $m = 1$ case, the shift in the second column makes the amplification of $\mathbf{s}^2$ at each iteration unity, etc. Geometrically, considering the convergence of the i'th column of V, we may draw a circle in the complex plane corresponding to the amplification of $\mathbf{s}^i$ due to the iteration; for sufficiently small h, the radius of this circle is close to unity, and the amplification of all remaining directions $\mathbf{s}^{i+1}$ through $\mathbf{s}^n$ are inside this circle. The small shift $(hV\Sigma)$ simply adjusts the radius of this circle to be exactly unity. Note that, though this refinement is not necessary in the standard subspace iteration approach, it is useful when we extend this approach in §4.2.

In summary, the iterative explicit subspace iteration algorithm, defined by (31b) or (36) [we use the latter], (32), (34b), (35a), and (35c), and implemented in an efficient order in Algorithm 2, converges (if h is sufficiently small) toward the leading components of the Schur decomposition

$$A\,V = V\,\Sigma, \tag{37}$$

determining a unitary $V = V_{n\times m}$ and upper-triangular $\Sigma = \Sigma_{m\times m}$, with λ_1 through λ_m on the main diagonal of Σ. The residual $r(i) = \mathbf{norm}(A\,V - V\Sigma)$ measures the degree to which (37) is not yet satisfied. Note that it is trivial to now compute the corresponding components of the eigen decomposition of A via the eigen

[12] Note, however, that those vectors still being worked on in V still need to be orthogonalized at every iteration against the converged Schur vectors that have been removed from the iteration [10].

decomposition (if it exists) of Σ:

$$\Sigma = \overline{S}\,\overline{\Lambda}\,\overline{S}^{-1} \quad \Rightarrow \quad AS = S\Lambda,$$

where $S = S_{n\times m} = V\overline{S}$ and $\Lambda = \Lambda_{m\times m}$ is diagonal.

4.1.2 Computational efficiency

Note finally that the orthogonalization and Schur decomposition steps in the main loop of Algorithm 2 may be made a bit more efficient by, instead of computing the $\underline{Q}\,\underline{R}$ decomposition of V, computing the Cholesky decomposition

$$Z = G\,G^H \quad \text{of the } (m\times m) \text{ matrix } \quad Z = V^H\,V, \tag{38a}$$

noting that, by construction, G is lower triangular. Also,

$$V = \underline{Q}\,\underline{R} \quad \Leftrightarrow \quad \underline{Q} = V\underline{R}^{-1}, \tag{38b}$$

and thus

$$V^H V = \underline{R}^H \underline{Q}^H \underline{Q}\;\underline{R} = \underline{R}^H \underline{R} \quad \Rightarrow \quad \underline{R} = G^H. \tag{38c}$$

As the solution of (37) is approached, it follows that

$$A(VG^{-H}) = (VG^{-H})\,(G^H\,\Sigma\,G^{-H}). \tag{38d}$$

We may now compute the Schur decomposition [cf. (35a)]

$$(G^H\,\Sigma\,G^{-H}) = \overline{U}\;\overline{T}\;\overline{U}^H, \tag{38e}$$

then update appropriately [cf. (35c)]

$$V \leftarrow V(G^{-H}\overline{U}) \quad \text{and} \quad \Sigma \leftarrow \overline{T}. \tag{38f}$$

After the computation of $(A\,V)$, the computational cost of which varies from problem to problem, the computational cost per iteration (when $n \gg m \gg 1$) following the $\underline{Q}\,\underline{R}$-based approach, as shown in Algorithm 2, is $\sim 8nm^2$ flops. In contrast, the Cholesky-based approach replaces the following two expensive steps:

- computation of the $V = \underline{Q}\,\underline{R}$ decomp. $\{\sim 2nm^2$ flops$\}$,
- computation of $V\,\overline{U}$ $\{\sim 2nm^2$ flops$\}$,

with the following two expensive steps:

- computation of $V^H V$ $\{\sim nm^2$ flops$\}$,
- computation of $V(G^{-H}\overline{U})$ $\{\sim 2nm^2$ flops$\}$.

Thus, neglecting the computational cost of the computation of $(A\,V)$, which is identical in the two approaches, the Cholesky-based approach is 12.5% cheaper ($\sim 7nm^2$ versus $\sim 8nm^2$).

4.1.3 Subspace iteration via IE discretization of (30)

Due to the restrictions on h inherent to the EE-based approach applied to stiff systems, we are also motivated to consider marching (30) with an Implicit Euler (IE) numerical discretization [cf. (31a)],

$$(I - h\,A)\mathbf{v}^{k+1} = \mathbf{v}^k, \tag{39a}$$

written again in a form that propagates a set of directions assembled as the columns of a matrix V [cf. (31b)],

$$(I - h\,A)V^{k+1} = V^k, \tag{39b}$$

which at each k may be solved for V^{k+1}. As the IE method is L stable [9], h may be made large without encountering numerical instability in this form of the march, which significantly improves the convergence of this IE-based form with k as compared with the EE-based form discussed previously. Note, however, that increasing h will also generally slow the convergence of an iterative solver used to solve (39b) for any given k, as it reduces the diagonal dominance of the matrix $(I - h\,A)$. Increasing h also has a diminishing effect on the convergence of the Schur vectors with k as h becomes large.

One (again, of many) options of approximating Σ at each step in this case is given by premultiplying (39b) by $(V^k)^H$, applying $(V^k)^H V^k = I$, and setting $A\,V^{k+1}$ equal to $V^{k+1}\Sigma^{k+1}$ [cf. (34b)], leading to:

$$\Sigma^{k+1} = \{I - [(V^k)^H V^{k+1}]^{-1}\}/h. \tag{39c}$$

As a refinement, we may apply a small shift $(hV\Sigma)$ to (39b) at each iteration [cf. (36)], thus marching

$$V^{k+1} - hA\,V^{k+1} = V^k - hV^k\Sigma. \tag{39d}$$

This has the same essential effect as marching (39b), with the benefit of ensuring that the update to V itself approaches zero as V approaches a basis of a set of Schur vectors, and thus $(A\,V - V\,\Sigma)$ approaches zero.

The other steps in the IE-based approach are analogous to those in the EE-based approach, and are implemented in Algorithm 3; note that $i_{\max}$ and ϵ define the stopping criterion of both algorithms. Note that an approximate form for the residual $r(i) = \mathbf{norm}(A\,V - V\,\Sigma)$ is implemented in this case in order to avoid an (expensive) computation of $A\,V$; strictly speaking, the approximate form implemented is valid only if h is sufficiently small that IE acts like EE, but this form is found to be adequate even for larger h, as it is only used to check convergence.

Algorithm 2 Prototype explicit subspace iteration algorithm to find the m least-stable eigenvalues (on the main diagonal of Σ), the corresponding Schur vectors (in V), and the corresponding eigenvectors (in S) of an $n \times n$ matrix A for $n \gg m \gg 1$.

```
determine $[n,n] = \mathbf{size}(A)$
set $m$ as the number of eigenvalues to compute
set $V = \mathbf{randn}(n,m)$, orthogonalize $V$
for $i = 1$ to $i_{\max}$ do
  set $L = A\,V$ ..............{flops: problem dependent}
  set $\Sigma = V^H L$ ........................{$\sim 2nm^2$ flops}
  set $L \leftarrow L - V\Sigma$ .....................{$\sim 2nm^2$ flops}
  set $r(i) = \mathbf{norm}(L)$
  set $V \leftarrow V + hL$
  compute $V = \underline{Q}\,\underline{R}$ decomposition, .....{$\sim 2nm^2$ flops}
    set $V \leftarrow \underline{Q}$, $\Sigma \leftarrow \underline{R}\,\Sigma\,\underline{R}^{-1}$
  compute $\Sigma = \overline{U}\;\overline{T}\;\overline{U}^H$ (Schur/real Schur) decomp.,
    set $\Sigma \leftarrow \overline{T}$, $V \leftarrow V\,\overline{U}$ ............{$\sim 2nm^2$ flops}
  if $r(i) < \epsilon$ then break end if
end for
{optional: compute eigenvectors $\overline{S}$ of $\Sigma$; set $S = V\,\overline{S}$}
```

Algorithm 3 Prototype implicit subspace iteration algorithm [cf. Algorithm 2].

```
determine $[n,n] = \mathbf{size}(A)$
set $m$ as the number of eigenvalues to compute
set $V = \mathbf{randn}(n,m)$, orthogonalize $V$
for $i = 1$ to $i_{\max}$ do
  solve $(I - hA)L = V$ for $L$ {flops: problem dependent}
  set $\Sigma = [I - (V^H L)^{-1}]/h$ ............{$\sim 2nm^2$ flops}
  set $L \leftarrow L - hL\Sigma$ ....................{$\sim 2nm^2$ flops}
  set $r(i) = \mathbf{norm}(L - V)/h$ ..{note: approximate form}
  compute $L = \underline{Q}\,\underline{R}$ decomposition, .....{$\sim 2nm^2$ flops}
    set $L \leftarrow \underline{Q}$, $\Sigma \leftarrow \underline{R}\,\Sigma\,\underline{R}^{-1}$
  compute $\Sigma = \overline{U}\;\overline{T}\;\overline{U}^H$ (Schur/real Schur) decomp.,
    set $\Sigma \leftarrow \overline{T}$, $V \leftarrow L\,\overline{U}$ .............{$\sim 2nm^2$ flops}
  if $r(i) < \epsilon$ then break end if
end for
{optional: compute eigenvectors $\overline{S}$ of $\Sigma$; set $S = V\,\overline{S}$}
```

4.2 Prototype OSSI algorithms

Our primary interest in subspace iteration methods in this paper is on how to extend such methods, as illustrated above in simplified prototype form, to find the *central* eigenvalues of a very large [e.g., $n \geq O(10^6)$] Hamiltonian matrix

$$Z = \begin{bmatrix} A & -BR^{-1}B^H \\ -Q & -A^H \end{bmatrix}. \tag{40a}$$

Such matrices arise, e.g., in optimal control problems [see (5a)], and have an eigenvalue structure that is symmetric across the imaginary axis.

The key idea is to determine the least-stable LHP Schur vectors V (i.e., the Schur vectors of Z corresponding to eigenvalues with negative real part that are closest to the imaginary axis), to partition these closed-loop Schur vectors into their state and adjoint components

$$V = \begin{bmatrix} X \\ P \end{bmatrix}, \tag{40b}$$

then to approximate, leveraging the Moore-Penrose pseudoinverse X^+ of the matrix X, the resulting feedback gain matrix [cf. (5f) and (6e)]:

$$K = -R^{-1}B^H(PX^+). \tag{40c}$$

The motivation for this idea is that, if the (neglected) closed-loop Schur vectors of Z are well damped, they likely play a reduced role in the full computation of K. Note that this idea was recently explored by [1,2] and found to be promising. Use of this closed-loop reduction technique might in practice, for $n \gg 1$, prove to be superior to the use of open-loop model reduction strategies, which typically fail to account for the control objective in the model reduction process.

Note again that, to date, *all* implementations of subspace iteration methods and their variants that we are aware of converge to *extremal* eigenvalues. Convergence to the *central* eigenvalues of Z (that is, those eigenvalues near the imaginary axis), using existing algorithms, requires a code that computes $Z^{-1}\mathbf{v}$ (or, via the Matrix Inversion Lemma, several calculations of $A^{-1}\mathbf{x}$), which is typically prohibitively slow. We thus seek a method to calculate the central Schur vectors of Z without access to the computation of either $Z^{-1}\mathbf{v}$ or $A^{-1}\mathbf{x}$. We have discovered a remarkably simple modification to Algorithms 2 and 3 which accomplishes this. To explain this modification, consider first what happens when, as in the MCE case, one of the off-diagonal terms of Z is zero. In this case, Algorithm 2 or 3 may be applied to compute

(a) the least-stable eigenvalues of A, and
(b) the least-stable eigenvalues of $-A^H$.

For the control of a system with a few unstable eigenvalues and many stable eigenvalues extending into the LHP, we need to know (a), which are the eigenvalues of A near the imaginary axis, but, rather than calculating (b), we instead need to find the *most*-stable eigenvalues of $-A^H$ (that is, the eigenvalues of $-A^H$ near the imaginary axis). It is a straightforward matter to find these eigenvalues simply by *changing the sign* of the related march of P (that is, in the march related to $-A^H$).

The eigenvalues of Z vary continuously as its elements are varied. If both Q and $BR^{-1}B^H$ are nonzero but the norm of their product is small (that is, a modest generalization from the MCE limit), application of a slightly modified form of Algorithm 2 or 3 to Z, as motivated above, returns those eigenvalues of Z near the least stable eigenvalues of A together with those eigen-

Algorithm 4 Prototype explicit oppositely-shifted subspace iteration (OSSI) algorithm for computing the least-stable of the LHP eigenvalues and the corresponding Schur vectors of a Hamiltonian matrix Z.

```
determine $[n,n] = \mathbf{size}(A)$
set $m$ as the number of eigenvalues of $Z$ to compute
set $X = \mathbf{randn}(n,m)$, $P = \mathbf{randn}(n,m)$
orthogonalize $X$
for $i = 1$ to $i_{\max}$ do
    set $X_1 = AX - (BR^{-1}B^H)P, \quad P_1 = -QX - A^H P$
    set $\Sigma = X^H X_1$
    set $X_1 \leftarrow X_1 - X\Sigma, \;\; P_1 \leftarrow P_1 - P\Sigma$
    set $r(i) = \mathbf{norm}\Big( \begin{bmatrix} X_1 \\ P_1 \end{bmatrix} \Big)$
    set $X \leftarrow X + hX_1$, $P \leftarrow P - hP_1$ {opposite shift!}
    compute $X = \underline{Q}\,\underline{R}$ decomposition,
        set $X \leftarrow \underline{Q}, \;\; P \leftarrow P\,\underline{R}^{-1}, \;\; \Sigma \leftarrow \underline{R}\,\Sigma\,\underline{R}^{-1}$
    compute $\Sigma = \overline{U}\;\overline{T}\;\overline{U}^H$ (Schur/real Schur) decomp.,
        set $\Sigma \leftarrow \overline{T}, \;\; X \leftarrow X\,\overline{U}, \;\; P \leftarrow P\,\overline{U}$
    if $r(i) < \epsilon$ then break end if
end for
```

values of Z near the most stable eigenvalues of $-A^H$, which are precisely the eigenvalues we seek.

4.2.1 OSSI via an EE discretization

The idea laid out above is implemented in Algorithm 4, which is essentially just Algorithm 2 applied to a matrix Z with the block structure given in (40a), with the sign modification on the shift discussed above and carefully-chosen formula implemented for the computation of Σ, discussed below. In principle, the update to X and P may be split into two parts, with a *positive* sign in the shift of X, and a *negative* sign in the shift of P:

$$X \leftarrow X + hX_1, \quad P \leftarrow P - hP_1 \tag{41a}$$

where

$$X_1 = AX - (BR^{-1}B^H)P, \quad P_1 = -QX - A^H P. \tag{41b}$$

To make such an iteration consistent, we may, in a manner analogous to that introduced in (36), apply small shifts to (41a) by marching

$$X \leftarrow X + h(X_1 - X\Sigma), \quad P \leftarrow P - h(P_1 - P\Sigma). \tag{41c}$$

This approach is referred to as *opposite shifting.*

As convergence is approached, $\{X, P\}$ approach a basis of the desired Schur vectors, and thus

$$ZV \approx V\Sigma \;\Leftrightarrow\; \begin{bmatrix} A & -BR^{-1}B^H \\ -Q & -A^H \end{bmatrix} \begin{bmatrix} X \\ P \end{bmatrix} \approx \begin{bmatrix} X \\ P \end{bmatrix} \Sigma. \tag{42}$$

As before, there are various options for computing Σ. A simple option which we have found to be effective is determined by multiplying the first block row of (42) times X^H and solving for Σ [that is, ignoring completely the second block row of (42)], resulting in

$$\Sigma = (X^H X)^{-1}(X^H X_1). \tag{43}$$

This option is thus analogous to (34b); note that the normalization factor $(X^H X)^{-1}$ may be skipped if X, rather than V, is orthogonalized via $\underline{Q}\,\underline{R}$ decomposition. This option is found in practice (see §5) to converge to the eigenvalues of Z just to the left of the imaginary axis, which are exactly those sought for the purpose of feedback control design [see (40)]. This method is implemented in Algorithm 4.

Algorithm 5 Prototype implicit oppositely-shifted subspace iteration (OSSI) algorithm [cf. Algorithm 4].

```
determine $[n,n] = \mathbf{size}(A)$
set $m$ as the number of eigenvalues of $Z$ to compute
set $X = \mathbf{randn}(n,m)$, $P = \mathbf{randn}(n,m)$, $\Sigma = \mathbf{zeros}(m,m)$
orthogonalize $X$
for $i = 1$ to $i_{\max}$ do
    solve $(I - hA)\,X_1 = X + h[-X\Sigma - (BR^{-1}B^H)P]$ for $X_1$,
          $(I - hA^H)\,P_1 = P - h[-P\Sigma - QX]$ for $P_1$
          ........................ {note opposite shift!}
    set $\Sigma \leftarrow \Sigma + X_1^H(X_1 - X)/h$
    set $r(i) = \mathbf{norm}\Big( \begin{bmatrix} X_1 \\ P_1 \end{bmatrix} - \begin{bmatrix} X \\ P \end{bmatrix} \Big)/h$
    compute $X_1 = \underline{Q}\,\underline{R}$ decomposition,
        set $X_1 \leftarrow \underline{Q}, \;\; P_1 \leftarrow P_1\,\underline{R}^{-1}, \;\; \Sigma \leftarrow \underline{R}\,\Sigma\,\underline{R}^{-1}$
    compute $\Sigma = \overline{U}\;\overline{T}\;\overline{U}^H$ (Schur/real Schur) decomp.,
        set $\Sigma \leftarrow \overline{T}, \;\; X \leftarrow X_1\,\overline{U}, \;\; P \leftarrow P_1\,\overline{U}$
    if $r(i) < \epsilon$ then break end if
end for
```

4.2.2 OSSI via an IE discretization

It is also straightforward to develop an implicit form of the OSSI algorithm described above, in an analogous manner to the development of the implicit form of the standard subspace iteration algorithm described in §4.1. In order to not invert Z and to leverage existing solvers for A, we actually only take the diagonal blocks of Z implicitly, and account for the off-diagonal blocks of Z explicitly. The update to Σ that we have chosen in this case is based on the relationship $dV/dt = ZV$ discretized via IE with a shift; setting $ZV_{n+1} = V_{n+1}\Sigma_{n+1}$, this results in

$$\frac{V_{n+1} - V_n}{h} = J(ZV_{n+1} - \underbrace{V_{n+1}\Sigma_n}_{\text{shift}}) = JV_{n+1}(\Sigma_{n+1} - \Sigma_n)$$

where $J = \big[I\ 0;\ 0\ -I\big]$. Looking at only the first block row of this result and premultiplying by X_{n+1}^H gives

$$\Sigma_{n+1} = \Sigma_n + (X_{n+1}^H X_{n+1})^{-1} X_{n+1}^H (X_{n+1} - X_n)/h.$$

Again, the normalization factor $(X^H X)^{-1}$ may be skipped if X is orthogonalized at each iteration via $\underline{Q}\,\underline{R}$ decomposition. This method is implemented in Algorithm 5.

5 Test on a representative LQR problem

A "representative" randomly-generated infinite-horizon LQR problem [see §1.1] may be created with A, $Q > 0$, $R = I$ and $\bar{B} = BR^{-1}B^H \geq 0$ defined via

$$\begin{aligned} A_1 &= \text{randn}(n,n); \quad A = -A_1\,A_1^H + \text{randn}(n,n);\\ Q_1 &= \text{randn}(n,n); \quad Q = \alpha\,Q_1\,Q_1^H;\\ \bar{B}_1 &= \text{randn}(n,m); \quad \bar{B} = \beta\,\bar{B}_1\,\bar{B}_1^H, \end{aligned}$$

where randn(n,m) denotes a random $n \times m$ matrix whose elements have zero mean and a Gaussian distribution with unit variance. The system matrix so created has both real and complex conjugate pairs of eigenvalues extending into the LHP, and usually has only a few unstable eigenvalues, which is typical in many problems of interest derived from well-posed PDEs. Several realizations of such randomly-generated LQR problems were studied when testing the methods described herein. All of the algorithms converge to the expected results when the parameters are selected appropriately (sufficiently small h, etc.). For a typical example, taking $n = 10$, $m = 4$, $\alpha = 0.1$, and $\beta = 0.01$, and approximately the optimal h in each case (found by a minor amount of trial and error), the convergence of Algorithms 2, 3, 4, & 5 are depicted in Figures 1 & 2.

Note that we have kept n small in the numerical tests reported here, as we have thus far only developed what we have referred to as "prototype" implementations of our new algorithms. Various standard acceleration techniques (deflating, implicit restarting, etc.) must be applied to the prototype subspace iteration algorithms reported here, in an analogous manner to how they have been applied to standard eigenvalue problems, before these new algorithms will be ready for application to control problems with large n (see, e.g., [15], and its numerical implementation in ARPACK).

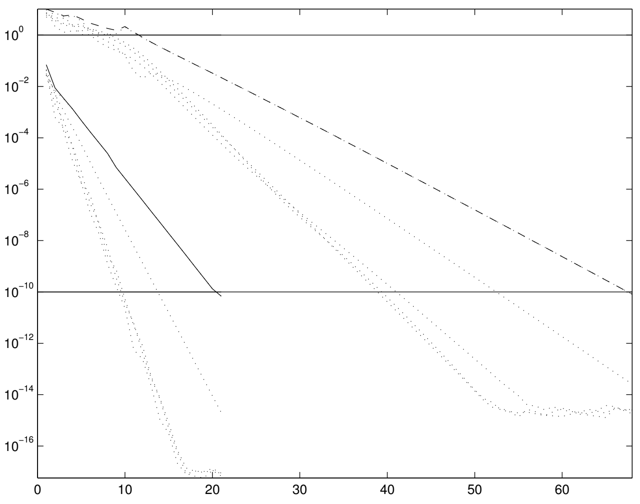

Fig. 1 Convergence of (dashed) the explicit form given in Algorithm 2, and (solid) the implicit form given in Algorithm 3, of the subspace iteration algorithm for the leading Schur vectors of the state matrix A. The dotted lines indicate the convergence of individual modes.

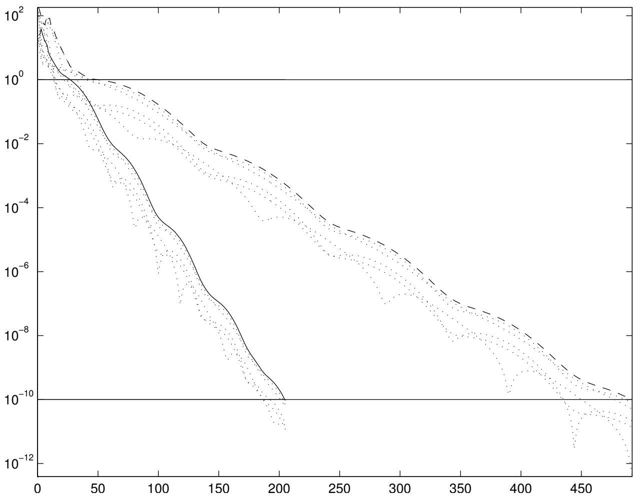

Fig. 2 Convergence of (dashed) the explicit form given in Algorithm 4, and (solid) the implicit form given in Algorithm 5, of the subspace iteration algorithm for the least-unstable LHP Schur vectors of the Hamiltonian matrix Z. The dotted lines indicate the convergence of individual modes.

6 Conclusions

This paper considers four methods for the efficient solution of optimal control problems for high-dimensional systems which bypass the intermediate and sometimes problematical step of open-loop model reduction. *Chandrasekhar's method*, reviewed in §1.2, is classical. The other three methods presented (MCE, ADA, OSSI) have been developed much more recently by our team.

The fact that the continuous-time stabilizing LQR controller in the *Minimum Control Energy* (*MCE*) limit simply reflects the unstable eigenvalues into the LHP is also classical. However, the algorithm reviewed in §2 to efficiently solve the LQR problem in the MCE limit in large-scale systems was apparently first identified only recently, in [14], and hasn't been widely recognized since. This algorithm requires only the eigenvalues and left eigenvectors of the unstable modes of the system matrix A, which may be determined via a subspace iteration method, such as those discussed in §4.1.

When the system of interest has open-loop eigenvalues that are near the imaginary axis, the performance of the controlled system in the MCE limit is sometimes inadequate, and a more aggressive control solution is desired. The remaining two approximate methods discussed in this paper provide attractive distinct alternatives to Chandrasekhar's method for the numerically-tractable approximate solution of high-dimensional optimal control problems outside of the MCE limit while bypassing open-loop model reduction.

The *Adjoint of the Direct-Adjoint* (*ADA*) algorithm of §3 was first introduced in [18]. It was also tested in [21] and [7]. Note also that the ADA method is a finite-horizon formulation (though the horizon selected may be taken as large), whereas MCE and OSSI are infinite-horizon formulations.

As mentioned previously, the utility of using the several least-stable of the stable Schur vectors of the Hamiltonian Z of the LQR problem in order to approximate the feedback gain K of the LQR problem was explored in some depth in [1,2]. The discovery of the efficient *Oppositely-Shifted Subspace Iteration* (*OSSI*) method of §4.2 to actually find these Schur vectors in high-dimensional systems is introduced for the first time in the present paper, and is motivated by the ideas behind the MCE and ADA methods.

The performance of explicit and implicit prototype OSSI algorithms on a model low-order control problem (see §5) is encouraging; their performance on high-dimensional discretizations of PDE control problems will be considered in future work.

As a matter of philosophy, the present work is the result of an integration of the perspectives of

(a) the motivating fluid-mechanical applications, which are characterized by significant eigenvector nonorthogonality and are thus not readily amenable to accurate model reduction for the purpose of feedback control design simply by retaining select controllable and observable eigenmodes,
(b) the optimization, control, and model reduction problems, and how they are related, and
(c) the advanced numerical methods required to approximate solutions to the problems in (b) for large n, and how they may be combined and extended.

The algorithms presented in this article for the feedback control of complex systems with $n \geq O(10^6)$ provide an attractive alternative to the two-step approach typically used, as described in the introduction. These algorithms arise, and may be further refined, only via an integration of three distinct and often unrelated traditional disciplines: system modeling, control theory, and numerical methods.

Conflict of Interest: The authors declare that they have no financial conflict of interest related to this research.

References

1. Amodei, L, & Buchot, J.-M. (2010) An invariant subspace method for large-scale algebraic Riccati equation. *Applied Numerical Mathematics* **60**, 1067-1082.
2. Amodei, L, & Buchot, J.-M. (2011) A stabilization algorithm of the Navier-Stokes equations based on algebraic Bernoulli equation *Numer. Linear Algebra Appl.*.
3. Anderson, BDO, & Moore JB (1971) *Linear Optimal Control.* Prentice Hall.
4. Bewley, TR (2001) Flow control: new challenges for a new Renaissance, *Progress in Aerospace Sciences* **37**, 21-58.
5. Bryson, AE, & Ho, Y-C (1969) *Applied optimal control.* Hemisphere.
6. Butler, KM, & Farrell, BF (1992) Three-dimensional optimal perturbations in viscous shear flow, *Phys. Fluids A* **4**, 1637-1650.
7. Carini, M, Pralits JO, Luchini, P (2015) Feedback control of vortex shedding using a full-order optimal compensator *J. Fluids Struct.* **53**, 15-25.
8. Golub, GH, & Van Loan, CF (1996) *Matrix Computations.* Johns Hopkins.
9. Hairer, E, & Wanner, G (1996) *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems*, 2nd edition, Springer-Verlag, Berlin.
10. Jennings, A, & Stewart, WJ (1981) A simultaneous iteration algorithm for real matrices, *ACM Trans. of Math. Software* **7**, 184-198.
11. Kailath, T (1973) Some New Algorithms for Recursive Estimation in Constant Linear Systems, *IEEE Trans. Information Theory* **19**, 750-760.
12. Kailath, T (1980) *Linear Systems.* Prentice-Hall.
13. Kim, J, & Bewley, TR (2007) A linear systems approach to flow control. *Annual Review of Fluid Mechanics* **39**, 383-417.
14. Lauga E, & Bewley, TR (2003) The decay of stabilizability with Reynolds number in a linear model of spatially developing flows. *Proc. R. Soc. Lond.* A **459**, 2077-2095.
15. Lehoucq, RB & Sorensen DC (1996) Deflation Techniques for an Implicitly Restarted Arnoldi Iteration, *SIAM J. Matrix Anal. & Appl.* **17**, 789-821.
16. Moore, BC (1981) Principal component analysis in linear systems: controllability, observability, and model reduction. *IEEE Trans. Autom. Control* **AC-26**, 1732.
17. Parlett, BN (1980) *The Symmetric Eigenvalue Problem.* Prentice Hall.
18. Pralits, JO, & Luchini, P (2009) Riccati-less optimal control of bluff-body wakes. *Proceedings of the Seventh IUTAM Symposium on Laminar-Turbulent Transition*, Stockholm, p. 325-330.
19. Rowley, CW (2005) Model reduction for fluids using balanced proper orthogonal decomposition. *International Journal of Bifurcation and Chaos* **15** (3), 9971013.
20. Saad, Y (1992) *Numerical Methods for Large Eigenvalue Problems.* Halstead Press, New York.
21. Semeraro, O, Pralits, JO, Rowley, CW, Henningson, DH (2013) Riccati-less approach for optimal control and estimation: An application in 2D Boundary Layers *J. Fluid Mech* **731**, 394-417.
22. Wilkinson, JH (1965) *The Algebraic Eigenvalue Problem.* Oxford.