On the numerical solution of a T-Sylvester type matrix equation arising in the control of stochastic partial differential equations

SUSANA N. GOMES
Department of Mathematics, South Kensington Campus, Imperial College London, SW7 2AZ, U.K
s.gomes12@imperial.ac.uk
AND
STEPHEN JAMES TATE
Department of Chemical Engineering, South Kensington Campus, Imperial College London, SW7 2AZ, U.K.
s.tate@imperial.ac.uk

Abstract

We outline a derivation of a nonlinear system of equations, which finds the entries of an \( m \times N \) matrix \( K \), given the eigenvalues of a matrix \( D \), a diagonal \( N \times N \) matrix \( A \) and an \( N \times m \) matrix \( B \). These matrices are related through the matrix equation \( D = 2A + BK + K^tB^t \), which is sometimes called a t-Sylvester equation. The need to prescribe the eigenvalues of the matrix \( D \) is motivated by the control of the surface roughness of certain nonlinear SPDEs (e.g., the stochastic Kuramoto-Sivashinsky equation) using nontrivial controls. We implement the methodology to solve numerically the nonlinear system for various test cases, including matrices related to the control of the stochastic Kuramoto-Sivashinsky equation and for randomly generated matrices. We study the effect of increasing the dimensions of the system and changing the size of the matrices \( B \) and \( K \) (which correspond to using more or less controls) and find good convergence of the solutions.

Keywords: Stochastic Kuramoto-Sivashinsky Equation, Matrix Equation, Control Theory.

1 Introduction

We present an algorithm for finding the entries of an \( m \times N \) matrix \( K \) related to the matrices \( D \), \( B \) and \( A \) by

\[
D = 2A + BK + K^tB^t, \tag{1.1}
\]

where \( t \) denotes the matrix transpose, the \( N \times N \) matrix \( D \) has prescribed eigenvalues \( \{\mu_i\}_{1 \leq i \leq N} \), \( A \) is a given diagonal \( N \times N \) matrix and \( B \) is a given \( N \times m \) matrix, with \( m < N \). This is a problem that arises when applying linear, point-actuated feedback controls to linear stochastic partial differential equations (SPDEs), as will be shown below.

The motivation for developing this algorithm is an application in control theory of nonlinear SPDEs such as the stochastic Kuramoto-Sivashinsky (sKS) equation,

\[
u_t = -\nu u_{xxxx} - u_{xx} - uu_x + \sigma \xi(x,t), \tag{1.2}
\]

which models the interface of systems that are part of surface roughening processes such as ion-sputtering processes or fluid flows influenced by thermal noise. The equation is considered in the periodic domain \( x \in (0,2\pi) \), \( \nu \) is an instability parameter, \( \xi(x,t) \) is a Gaussian noise with mean zero and a given covariance, and \( \sigma > 0 \) is its strength. We note that when \( \nu < 1 \), there are positive eigenvalues associated with the eigenfunctions \( \sin(kx), \cos(kx) \) of the linear operator \( -\nu \partial_{xxxx} - \partial_{xx} \) for \( k < \frac{1}{\sqrt{\nu}} \).
In surface roughening processes, it is of particular interest to monitor and control the surface roughness of the interfaces, and this is measured by the $L^2$-norm of its fluctuations around a steady state $\bar{u}$ (usually $\bar{u}(x,t) = 0$). The control of the sKS equation using nonlinear distributed feedback controls, i.e., controls that are applied everywhere in the physical domain, and depend nonlinearly on the solution at every time step, was studied by Lou, Christofides and collaborators (see, e.g., Lou (2005)). There, the authors essentially add controls that are proportional to $v u_{xxxx} + u_{xx} + u u_t$ and modify the dynamics so that the SDE becomes linear. Recently, Gomes et al. (2017) focussed on the case when the feedback controls are linear, and its application is point-actuated. In this case, the authors use a splitting method, which allows to separate the SDE into a nonlinear deterministic PDE (with random coefficients)

$$v_t = -v v_{xxxx} - v_{xx} - (v w)_x - w w_x, \tag{1.3}$$

which is coupled to a linear SDE

$$w_t = -v w_{xxxx} - w_{xx} + \sigma \xi(x,t). \tag{1.4}$$

In this paper, we focus on equation (1.4). This is because with the appropriate choice of controls applied to the deterministic equation (1.3), one can guarantee that $v = 0$ and therefore we are able to control the surface roughness of the solution $u$ to (1.2) by controlling the surface roughness of $w$. The discretisation of this problem leads to the necessity to solve equation (1.1), as will be shown in Section 2.

This paper is organised as follows. In Section 2 we outline the derivation of equation (1.1), make an overview of the relevant literature, and explain why the existing static output feedback control and Sylvester equation solvers are not enough to solve the problem. Section 3 will be concerned with the derivation of the system of nonlinear equations for the entries of $K$, and in Section 4 we solve this system of equations using standard optimisation techniques, for randomly generated matrices, as well as for the linear sKS equation. We close with conclusions and further work motivated by our algorithm in Section 5.

## 2 Outline of the derivation of equation (1.1)

Consider the linear sKS equation under the influence of point-actuated controls

$$w_t = -v w_{xxxx} - w_{xx} + \sigma \xi(x,t) + \sum_{i=1}^{m} b_i(x)f_i(t), \tag{2.1}$$

in the domain $x \in [0,2\pi]$, with periodic boundary conditions, and initial condition $w(x,0) = w_0(x)$. Furthermore, we assume that the solution $w$ has mean zero. Here, $v$ is the instability parameter, $m$ is the number of controls, $b_i(x)$, $i = 1, \ldots, m$ are the point actuators (we consider $b_i(x) = \delta(x-x_i)$, where $\delta(\cdot)$ is the Dirac delta function) and $f_i(t)$ are the control amplitudes, which we will choose to be a linear function of the solution $w$ (more precisely, of its Fourier series coefficients as will become clear below).

When we write the linear sKS equation in Fourier space, the linear operator $-v \partial_{xxxx} - \partial_{xx}$ is represented by the matrix $A = \text{diag}(a)$, where $a_{2k-1} = a_{2k} = -vk^4 + k^2$ for $k = 1,2,\ldots$, and the control actuators are represented by $B = [b_{jk}]$, where $b_{(2j-1)k} = \int_0^{2\pi} \delta(x-x_k) \sin(jx) \ dx = \sin(jx_k)$ and $b_{(2j)k} = \int_0^{2\pi} \delta(x-x_k) \cos(jx) \ dx = \cos(jx_k)$. We truncate the system at $N$ Fourier modes and by noting that $\{\sin(jx), \cos(jx)\}_{j \in \mathbb{N}}$ is a basis of $L^2_p(0,2\pi)$, the space of periodic functions of zero mean in $(0,2\pi)$, we expect that, for $N$ large enough, the discretised version is a good approximation to the continuous version.

Denoting by $w = [w_1, \ldots, w_N]'$ and $\xi = [\xi_1, \ldots, \xi_N]'$ the vectors of Fourier coefficients of $w$ and $\xi$ respectively, the controlled system can be discretised in the form $w_t = Aw + Bf + \sigma \xi$, where we defined the controls as a linear function of the Fourier coefficients of $w$, $F = Kw$. In summary, we need to solve the linear stochastic PDE

$$w_t = (A + BK)w + \sigma \xi = Cw + \sigma \xi. \tag{2.2}$$

We note that due to the nature of the matrix $B$ this system is fully coupled, which is the source of the difficulty on the solution to this problem.
As was noted above, the goal of Gomes et al. (2017) is to control the expected value of the surface roughness of the solution $w$ to the linear sKS equation. The surface roughness is defined as follows.

$$r(t) = \sqrt{\frac{1}{2\pi} \int_0^{2\pi} (w(x,t) - \bar{w}(t))^2 \, dx},$$

(2.3)

where $\bar{w}(t) = \int_0^{2\pi} w(x,t) \, dx$ (which is zero in our case). This quantity is known to saturate for large times (Barabasi and Stanley, 1995), and we wish to control its saturated value. We can control the long-time behaviour of the surface roughness by controlling the expected value of its square. Considering the Fourier series representation of $w$ and denoting the expected value of a function by $\langle \cdot \rangle$, we can write this quantity as

$$\langle r(t)^2 \rangle = \sum_{k=1}^{N/2} \langle w_{2k-1}^2(t) \rangle + \langle w_{2k}^2(t) \rangle,$$

(2.4)

which, in turn, is the trace of the covariance matrix of the solution $w$, $Q(t)$, defined as $Q_{ij}(t) = \langle w_i(t)w_j(t) \rangle$.

For a decoupled system (i.e., if $B$ and $K$ are diagonal matrices), the solution of this problem is straightforward (see Gomes et al. (2017) or Hu et al. (2008)). However, when the matrix $B$ is full, the closed loop system is not decoupled and therefore standard ways of obtaining the second moment of the coefficients are not applicable here. We (see Gomes et al. (2017) for more details on this derivation) instead use Theorem 4 in Jimenez (2015) to find that the covariance matrix $Q(t)$ of the solution $w$ to $w_t = Cw + \sigma r$, is given by

$$Q(t) = H_1 F_1 + F_1 H_1,$$

where $F_1$ and $H_1$ are the $(1,1)$ and $(1,3)$ blocks of the matrix $e^{Mt}$, respectively. Here,

$$M = \begin{bmatrix}
C & 0 & \frac{2}{3}I & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & -C' & 0 \\
0 & 0 & 0 & 0
\end{bmatrix},$$

$I$ is the $N \times N$ identity matrix and the zeros stand for appropriately sized zero matrices. We compute $e^{Mt}$ and conclude that $F_1 = e^{Ct}$, and

$$H_1 = \frac{\sigma^2}{2} \left[ I_t + (C+C') t^2 / 2 + (C^2 + 2CC' + (C')^2) t^3 / 3! + (C^3 - C^2C' + C(C')^2 - (C')^3) t^4 / 4! + \cdots \right],$$

from where we obtain

$$Q(t) = \sigma^2 \left[ I_t + (C+C') t^2 / 2 + (C^2 + 2CC' + (C')^2) t^3 / 3! + (C^3 + 3C^2C' + 3C(C')^2 + (C')^3) t^4 / 4! + \cdots \right].$$

(2.5)

The problem now is that due to the nature of the matrix $B$, $C$ is not normal, i.e. it does not commute with its transpose. However, we are not interested in controlling the full matrix $Q(t)$ but only its trace. Making use of the linearity of the trace and its continuity, as well as identities of the type

$$\text{tr} \left( C^2 + 2CC' + (C')^2 \right) = \text{tr} \left( C^2 + CC' + C'C + (C')^2 \right) = \text{tr} (C + C')^2,$$

we obtain

$$\text{tr}(Q(t)) = \sigma^2 \left( \text{tr}(I) + \text{tr}(C+C') t^2 / 2 + \text{tr} (C+C')^2 t^3 / 3! + \text{tr} (C+C')^3 t^4 / 4! + \text{tr} (C+C')^4 t^5 / 5! + \cdots \right).$$
We assume that $C + C'$ is invertible so that we can multiply by $I = (C + C')^{-1}(C + C')$ and add and subtract adequate terms to obtain

$$\text{tr}(Q(t)) = -\sigma^2 \text{tr}\left[ (C + C')^{-1} \right] + \sigma^2 \text{tr}\left( (C + C')^{-1} \sum_{n \in \mathbb{N}} (C + C')^n \frac{r^n}{n!} \right).$$  
(2.6)

Moreover, if we guarantee that all of the eigenvalues of $D = C + C'$ are negative, then as $t \to \infty$ we obtain that the saturated value of the surface roughness is

$$r_d^2 = \lim_{t \to \infty} < r(t)^2 > = -\sigma^2 \text{tr}(D^{-1}).$$

**Remark 2.1** We note that Equation (2.2) is a multidimensional Ornstein-Uhlenbeck process, and therefore we could use an alternative method for computing its covariance matrix by applying Itô isometry, obtaining

$$Q(t) = \int_0^t e^{C^T s} \sigma \sigma^T e^{C s} ds.$$  
(2.7)

Using the fact that $\sigma$ is scalar and the dominated convergence theorem would give us the Taylor expansion (2.5) directly. Furthermore, taking the limit $t \to \infty$ in (2.7), it follows that $Q = \lim_{t \to \infty} Q(t)$ solves the Lyapunov equation

$$CQ + QC^T = -\sigma \sigma^T,$$

from where we can also deduce the expression for $r_d^2$.

Hence, we need to find a matrix $K$ so that the trace of the matrix $D^{-1}$ is prescribed. This can be obtained by controlling the eigenvalues of $D$. Using the definition of $D$, we observe that we wish to solve the matrix problem (1.1). As far as we are aware, there is no known solution to this equation. The usual pole placement (Kautsky et al., 1985; Yang & Orsi, 2006) or LQR (Zabczyk, 1992) algorithms well known in control theory are only valid to either stabilise the matrix $C$, i.e., make all of its eigenvalues negative, or prescribe the eigenvalues of $C$. These are not useful here, since prescribing the eigenvalues of $C$ gives us, at best, a loose estimate on the eigenvalues of $D$.

Another possibility would be to use algorithms for static output feedback pole placement for symmetric systems, where the matrix $K$ is constrained to be of the form $SB$ for a symmetric $m \times m$ matrix $S$. These algorithms are designed to prescribe the eigenvalues of $C = A + BK$ and since $C$ is symmetric in this case, this also prescribes the eigenvalues of $D = 2(A + BSB')$. Algorithms for the numerical solution of this problem were proposed in Mahony et al. (1993) and Yang & Orsi (2006); our methodology is more general in that it does not require $C$ to be symmetric.

Mahony et al. (1993) reformulates the problem as a least squares problem and derive two gradient flow solutions which are based on the solution of ODEs for the matrix $S$ and a gradient flow, $\Theta$. The equilibrium solutions (which are shown to exist) of this system of ODEs define a minimizer of the least squares problem. In Yang & Orsi (2006) the authors use an iterative algorithm based on alternate projection ideas. Their method works for the general problem, and considers several problems, namely classic pole placement (where they prescribe the eigenvalues of $C$), stabilization (where the only requirement is that eigenvalues of $C$ have negative real part), relaxed pole placement (where the eigenvalues are required to lie in a disc with a prescribed center and radius), and hybrid problems (where the eigenvalues are required to belong to some prescribed region of the complex plane). This methodology is not guaranteed to find a solution, but numerical experiments show that the methodology is effective in practice.

There are many other control-related matrix equation problems which have been posed and solved. One example is the Sylvester equation (Sylvester, 1884), for which we need to find the matrix $X$ solving

$$AX + XB = C,$$  
(2.8)

for given matrices $A$, $B$ and $C$. A further example is the famous Lyapunov equation, where an algorithm was given by Bartels & Stewart (1972), where the aim is to solve

$$AX + XA^T = C,$$  
(2.9)
There are well known sufficient conditions for existence of solutions for these equations. None of these, however, can be adapted to solve equation (1.1).

An equation of the form

$$AX + X'B = C$$  \hfill (2.10)

is sometimes called T-Sylvester equation. The solution to this equation with $C = 0$ was recently studied in De Terán et al. (2013), where the authors describe how to find the general solution for arbitrary complex matrices $A \in \mathbb{C}^{m \times n}$ and $B \in \mathbb{C}^{n \times m}$. They show that the solution can be obtained in terms of the matrix pencil $A + \lambda B^*$ and matrices related to the pencil’s Kronecker canonical form, and give a complete description of the solution as long as these matrices are known. However, development of numerical methods to find the general solution of $AX + X'B = C$ is consistent and computing its general solution is still an open problem.

Braden (1998) gives necessary and sufficient conditions for the equation

$$AX \pm X'A = B$$  \hfill (2.11)

to admit solutions and gives the general solution for $X$, where $B$ is a (given) arbitrary square matrix. In this paper, $A$ needs not be square or invertible, and therefore the method uses pseudo-inverses (i.e., matrices $G$ such that $AGA = A$) only. The author proves that Equation (2.11) has solutions if and only if $B' = \pm B$ and $(1 - P_1)B(1 - P_1) = 0$, where $P_1$ is a projection operator, and gives a general solution in terms of $B, P_1, A, G$ and arbitrary matrices satisfying certain symmetry requirements.

Equation (1.1) differs from (2.11) and (2.10) in that it has an extra term, the matrix $D$, whose entries are not specified, and the only requirement is that it has certain eigenvalues. It is, therefore, an additional unknown in the problem, which makes our problem more general. If one decides to fix $D$ such that $D - 2A$ is symmetric (or anti-symmetric) and satisfies the second condition of (Braden, 1998, Thm1), then we can use the general solution derived there. However, this would require yet another solution of a matrix equation for $D$.

### 3 The nonlinear system of equations for the entries of $K$

In this section, we will derive a system of nonlinear equations for the entries of the $m \times N$ matrix $K$, given the $N \times N$ diagonal matrix $A = \text{Diag} (\{a_i\}_{1 \leq i \leq N})$, the eigenvalues of the matrix $D$, $\{\mu_l\}_{1 \leq l \leq N}$, and the entries of the $N \times m$ matrix $B$. We first write this system of equations, distinguishing the case $m = 1$ and $m > 1$ and then proceed to the derivation of the equations.

In what follows, we use the notation $\text{det}_J(M)$ for the diagonal minor of the matrix $M$ corresponding to the entries in the subset $J \subseteq \{1, \ldots, N\}$, i.e., the determinant of the submatrix of $M$ formed from the rows and columns corresponding to the index set $J$.

For the case $m = 1$, we have to solve the system of nonlinear equations:

$$\sum_{J \subseteq [N]} \prod_{\{j\} \in J} \mu_j = 2^J \sum_{J \subseteq [N]} \prod_{\{j\} \in J} a_j \left(1 + \sum_{p \in J} b_{p} k_{p} + \sum_{p \in J} \sum_{q \in J \setminus \{p\}} b_{q} k_{q} \frac{b_{p} k_{p}}{4} \Delta_{p, q} \right),$$  \hfill (3.1)

for $l = 1 \ldots N$. Here we have denoted for convenience the entries of the column vector $B$ as $b_q$ and the entries of the row vector $K$ as $k_q$.

For the case $m \geq 2$, we have the system of equations

$$\sum_{J \subseteq [N]} \prod_{\{j\} \in J} \mu_j = 2^J \sum_{J \subseteq [N]} \prod_{\{j\} \in J} a_j \left(\prod_{J \subseteq [N]} \frac{\text{det}_J(M)}{\Delta_{m, J}}\right),$$  \hfill (3.2)
where

\[ \Delta_{(m,J)} := \det (\delta_{ij} + \eta_{ij})_{1 \leq i,j \leq m} \]  
(3.3)

\[ \eta_{ij}^{(J)} := \sum_{l \in J} \frac{k_l b_{lj}}{2a_l} \]  
(3.4)

\[ \gamma_{ij}^{(m,J)} := \Delta_{(m,J)}(\delta_{ij} + \eta_{ij}) - \sum_{p,q=1}^{m} \beta_{ij}^{(p)} \tau_{pq}^{(J)} \kappa_{ij}^{(J)} \]  
(3.5)

\[ \beta_{ij}^{(J)} := \sum_{l \in J} b_{il} b_{lj} \frac{1}{2a_l}, \quad \kappa_{ij}^{(J)} := \sum_{l \in J} \frac{k_l \beta_{lj}}{2a_l}, \quad \tau_{pq}^{(J)} := (\adj (\delta_{ij} + \eta_{ij}))_{pq} \]  
(3.6)

### 3.1 The key idea behind the derivation

We wish to prescribe only the eigenvalues for \( D \) and therefore we need only the information provided by the characteristic polynomial of \( D, \chi_D(t) \). We have that

\[ \chi_D(t) = \prod_{i=1}^{N} (t - \mu_i) = \prod_{L=0}^{N} (-1)^L \sum_{J \mid |J| = L} \prod_{j \in J} \mu_j t^{N-L}. \]  
(3.7)

In general, the characteristic polynomial of a matrix \( M \) can be expressed in terms of the sum over all diagonal minors of size \( l \), denoted \( \sigma_l \), by:

\[ \chi_M(t) = \sum_{l=0}^{N} (-1)^l \sigma_l t^{N-l}. \]  
(3.8)

Therefore, we need to calculate the diagonal minors of \( 2A + BK + K' B' \). For the individual subcases this is done in the following subsections. Once the diagonal minors are calculated, we obtain the set of \( N \) equations

\[ \sum_{J \mid |J| = l} \prod_{j \in J} \mu_j = \sigma_l = \sum_{J \mid |J| = l} \det_J (2A + BK + K' B'), \]  
(3.9)

where \( \det_J \) denotes the diagonal minor of the matrix formed from the rows and columns corresponding to the index set \( J \).

### 3.2 The key Lemmas

The nonlinear system of equations in terms of the entries of \( K \) is derived by using two key lemmas. For each lemma, we have a special formula for the \( m = 1 \) case. The matrix determinant lemma is found in Press et al. (1972)[p. 73] and is applied to the Google matrix for a more specialised problem in Ding & Zhou (2007).

**Lemma 3.1 (Matrix-Determinant Lemma)** For an invertible \( N \times N \) matrix \( A, N \times m \) matrix \( B \) and \( m \times N \) matrix \( K \), where \( m < N \), we have the following identity

\[ \det (A + BK) = \det (A) \det (I_m + KA^{-1}B). \]  
(3.10)

The above lemma can be written in terms of column vectors \( u \) and \( v \) in the case \( m = 1 \) and may be written without determinants.

**Corollary 3.1 (Matrix Determinant Lemma - Rank One Version)** For an invertible \( N \times N \) matrix \( A \) and \( N \times 1 \) vectors \( u \) and \( v \), we have the following identity:

\[ \det (A + uv^T) = \det (A)(1 + v^T A^{-1} u). \]  
(3.11)

If \( A \) is not invertible, then we have the following version:

\[ \det (A + uv^T) = \det (A) + v^T \adj (A) u. \]  
(3.12)
We have a slightly complicated inverse to calculate in our particular application and so we employ the Woodbury identity (Woodbury, 1950).

**Lemma 3.2 (Woodbury Matrix Identity)** For an \( N \times N \) invertible matrix \( A \), \( N \times m \) matrix \( B \) and \( m \times N \) matrix \( K \), we have the following identity:

\[
(A + BK)^{-1} = A^{-1} - A^{-1}B(I_m + KA^{-1}B)^{-1}KA^{-1},
\]

as long as \( I_m + KA^{-1}B \) is invertible.

There is a specialisation to rank one updates to inverse matrices called the Sherman Morrison formula (Sherman & Morrison, 1950). It is noted that this formula appeared earlier in the literature by Hager (Hager, 1989).

**Corollary 3.2 (Sherman Morrison Formula)** Let \( A \) be an \( N \times N \) invertible matrix and \( u \) and \( v \) be \( N \times 1 \) vectors, and suppose that \( 1 + v' A^{-1} u \neq 0 \). Then we have the following expansion of the rank one perturbation of the inverse of the matrix \( A \):

\[
(A + uv')^{-1} = A^{-1} - \frac{A^{-1}uv'A^{-1}}{1 + v'A^{-1}u}.
\]

We appropriately extend these lemmas for the purposes of our problem.

**Lemma 3.3 (Symmetric Extension of Matrix Determinant Lemma)** Let \( A \) be an invertible \( N \times N \) matrix, \( B \) be an \( N \times m \) matrix and \( K \) an \( m \times N \) matrix, be such that \((A + BK)^{-1}\) and \((I_m + K(2A)^{-1}B)^{-1}\) exist. Then we have the following identity:

\[
\det (2A + BK + K'B') = \det (2A)\det (I_m + K(2A)^{-1}B)\det (I_m + B'(2A)^{-1}K')
\]

\[
- B'(2A)^{-1}B(I_m + K(2A)^{-1}B)^{-1}K(2A)^{-1}K'.
\]

**Proof.** We apply the Matrix-Determinant Lemma (Lemma 3.1) to \( \det (2A + BK + K'B') \), factorising out \( 2A + BK \):

\[
\det (2A + BK + K'B') = \det (2A + BK)\det (I_m + B'(2A + BK)^{-1}K').
\]

We then apply the Matrix Determinant lemma (Lemma 3.1) to the first determinant, to take out a factor of \( \det (2A) \):

\[
\det (2A + BK) = \det (2A)\det (I_m + K(2A)^{-1}B)\det (I_m + B'(2A + BK)^{-1}K').
\]

We now have the first two factors and we need to use the Woodbury Matrix Identity (Lemma 3.2) to expand the inverse in the third determinant around \( 2A \):

\[
B'(2A + BK)^{-1}K' = B'(2A)^{-1}K' - B'(2A)^{-1}B(I_m + K(2A)^{-1}B)^{-1}K(2A)^{-1}K'.
\]

We then combine (3.18) with (3.17) to obtain the required expression.

In the rank one case, we may use the Sherman Morrison formula and obtain a simple corollary.

**Corollary 3.3 (Symmetric Extension of Matrix Determinant Lemma - Rank One Version)** For an invertible and symmetric \( N \times N \) matrix \( A \) and \( N \times 1 \) vectors \( u \) and \( v \) such that \( 2A + uv' \) is invertible and \( 1 + v'(2A)^{-1}u \neq 0 \), we have the following identity:

\[
\det (2A + uv' + vu') = \det (2A) \left( (1 + v'(2A)^{-1}u)^2 - v'(2A)^{-1}uv'(2A)^{-1}v \right)
\]

**Proof.** This identity follows from repeated application of Lemma 3.1 as follows:

\[
\det (2A + uv' + vu') = \det (2A + uv')(1 + u'(2A + uv')^{-1}v)
\]

\[
= \det (2A)(1 + v'(2A)^{-1}u)(1 + u'(2A + uv')^{-1}v).
\]
We then apply the Sherman Morrison formula to the last bracket to obtain
\[
\det (2A + u v^t + v u^t) = \det (2A) (1 + v^t (2A)^{-1} u) \times \\
\times \left( 1 + u^t (2A)^{-1} v - \frac{v^t (2A)^{-1} u v^t (2A)^{-1} v}{1 + v^t (2A)^{-1} u} \right). \tag{3.21}
\]

We realise that \(v^t (2A)^{-1} u = u^t (2A)^{-1} v\), since \((2A)^{-1}\) is symmetric and the transpose of a scalar is the same scalar. We therefore multiply through by the second factor to obtain the expression
\[
\det (2A + u v^t + v u^t) = \det (2A) \left( (1 + v^t (2A)^{-1} u)^2 - v^t (2A)^{-1} u v^t (2A)^{-1} v \right). \tag{3.22}
\]

\[\square\]

### 3.3 Derivation for the case \(m = 1\)

We use vector notation for \(B\) and \(K\) and denote them by \(u\) and \(v\) respectively. This is to emphasise that we are dealing with a column vector and a row vector. In this case, we may rewrite (1.1) as
\[
D = 2A + uv^t + vu^t. \tag{3.23}
\]

To calculate \(\det_j(2A + uv^t + vu^t)\), we use Corollary 3.3, to write it as
\[
\det_j(2A) \left( (1 + v^t (2A)^{-1} u)^2 - u^t (2A)^{-1} u v^t (2A)^{-1} v \right). \tag{3.24}
\]

The matrix products are all done with index set \(J\) and so multiplying out this expression and writing it in component form gives
\[
\prod_{i \in J} (2a_j) \left( 1 + \sum_{j \in J} v_j u_j a_j + \sum_{j \in J} \frac{v_j v_k}{a_j a_k} (v_j u_k - u_j v_k) \right). \tag{3.25}
\]

To obtain \(\sigma_k\), we sum over all index sets of size \(k\) and make the identification \(B = u\) and \(K = v\) in order to obtain the expression (3.1).

### 3.4 The case \(m \geq 2\)

To derive the nonlinear system (3.2), we need to calculate \(\det_j(2A + BK + K' B')\) and so we use the Symmetric Extension of the Matrix Determinant Lemma (Lemma 3.3) to obtain the expression
\[
\det_j(2A + BK + K' B') = \det_j(2A) \det (I_m + (2A)^{-1} B) \det (I_m + B' (2A)^{-1} K') \\
- B' (2A)^{-1} B (I_m + (2A)^{-1} B)^{-1} K (2A)^{-1} K', \tag{3.26}
\]

where the indices of matrix multiplication made inside these determinants are from \(j \in J\).

Recalling the definition of \(\eta_{ij}^{(J)}\), we realise the second determinant is precisely \(\Delta_{(m,J)}\) and that the first two terms in the third determinant are \(\delta_j + \eta_{ji}\), when written in index form. It is also evident from the definitions of \(\beta_{ij}^{(J)}\), \(\kappa_{ij}^{(J)}\) and \(\tau_{ij}^{(J)}\) in equation (3.6), along with Cramer’s rule that the final entry in the third determinant is
\[
\sum_{p,q=1}^m \beta_{ij}^{(J)} \kappa_{ij}^{(J)} \tau_{ij}^{(J)} \Delta^{-1}_{(m,J)}.
\]

We therefore factorise \(\frac{1}{\Delta_{(m,J)}}\) out of the determinant, to get the expression
\[
\det_j(2A + BK + K' B') = 2^{\left|J\right|} \prod_{j \in J} a_j \frac{\det (T_{pq}^{(J)})}{\Delta^{-1}_{(m,J)}}, \tag{3.27}
\]

where \(T_{pq}^{(m,J)}\) is as described in equation (3.5).
4 Implementation and numerical results

In order to solve the matrix equation (1.1), we need to solve a nonlinear system of equations for the entries of the $m \times N$ matrix $K$. When $m = 1$, system (3.1) is a system of $N$ equations for $N$ unknowns, in which case we can use a standard Newton solver in MATLAB. For $m > 1$ the system (3.2) has $N$ equations for $mN$ unknowns and therefore it is underdetermined. In this case, the system may have no solutions, one solution, multiple ones or even infinitely many - we note that in the $m = 1$ case there is also the possibility that a solution does not exist. We use MATLAB’s nonlinear solver to solve the desired system of equations, and for under- and overdetermined systems, MATLAB uses the Levenberg-Marquardt algorithm (Levenberg, 1944; Marquardt, 1963). This is an algorithm that solves nonlinear least squares problems by interpolating between the Gauss-Newton and gradient descent methods. It is a more robust method than the application of Gauss-Newton by itself, which guarantees that in most cases a solution will be found, even if the starting point is far from a minimum. However, there is the possibility that the solution found is a local minimum, and not necessarily a global one.

We tested the efficiency of the algorithm described here for both $m = 1$ and $m > 1$ and for various values of $N$ and different matrices $A$ and $B$. We define two types of error for this problem. The first type of error we consider, $E_1$, is defined by how far are the obtained eigenvalues $\mu_k$ from the prescribed ones ($\mu_k^\text{pres}$):

$$E_1 = \sqrt{\sum_{k=1}^{N} (\mu_k - \mu_k^\text{pres})^2}.$$  

$E_1$ is small if we find a solution to the system of nonlinear equations. However, since there is a possibility that solutions do not exist, we consider a second type of error, which measures how far the obtained trace is from the desired trace, $\sum_{k=1}^{N} \mu_k$:

$$E_2 = \left| \text{Tr}(2A + BK + K'B) - \sum_{k=1}^{N} \mu_k \right|.$$  

We expect $E_2$ to be small independently of whether the system has an exact solution or not, since one of the $N$ equations (the one for $|J| = 1$) corresponds precisely to $E_2$ being zero.

4.1 Control of the stochastic Kuramoto-Sivashinsky equation

We first apply our algorithm to our motivating example, the linear sKS equation. In this application, we wish to maintain the negative eigenvalues and change the positive ones to a value that will allow the prescribed trace for the matrix $D$. For the purpose of this example, we will choose

$$\mu_j = \begin{cases} 2a_j & \text{if } a_j < 0, \\ -a_j & \text{if } a_j > 0, \\ -0.1 & \text{if } a_j = 0. \end{cases} \quad (4.1)$$

We used $\nu = 0.5$, which has two positive entries in the matrix $A$. In this case, it makes sense to use $m = 2$ and we chose $x_1 = \pi/10$ and $x_2 = 5\pi/3$. The convergence results for the sKS equation are as summarised below in Table 1, where we describe the running time, number of iterations, function evaluation count and errors $E_1$ and $E_2$ as a function of $N$, the size of the system. We note that $N$ is even, due to the nature of the eigenvalues of the original problem.

| $N$   | 2    | 4    | 6    | 8    | 10   | 12   | 14   |
|-------|------|------|------|------|------|------|------|
| Run time (s) | 0.05 | 0.16 | 1.09 | 4.29 | 23.30| 97.43| 495.13|
| Nr of iterations | 10   | 16   | 24   | 18   | 17   | 17   | 18   |
| Function Count  | 57   | 162  | 345  | 349  | 415  | 499  | 614  |
| $E_1$   | 0.0014 | 0.0688 | 0.1217 | 0.0933 | 0.0880 | 0.0784 | 0.1047 |
| $E_2$   | $2.6 \times 10^{-9}$ | $2.6 \times 10^{-4}$ | 0.0055 | 0.0110 | 0.0165 | 0.0349 | 0.0144 |

Table 1: Results of the numerical method for the solution of the matrix problem for the sKS equation with $\nu = 0.5$. 


4.2 Control of randomly generated matrices

To test the validity of our algorithm in more general cases, we generate a vector \( a \) with \( N \) entries randomly distributed, following a Gaussian law with mean zero and variance 1. We define the initial \( \mu_0 = 2a \) and change \( 1, 2, \ldots, 5 \) of its elements to obtain the \( \mu \) that we require. We then use \( m = 1, 2, \ldots, 5 \), respectively, to obtain the matrix \( K \) that gives us the desired eigenvalues. We specify below the randomly generated matrix \( B \) and vector \( a \) we use in Tables 2-9.

\[
\begin{bmatrix}
0.5377 \\
1.8339 \\
-2.2588 \\
0.8622 \\
0.3188 \\
-1.3077 \\
-0.4336 \\
0.3426 \\
3.5784 \\
2.7694 \\
-1.3499 \\
3.0349 \\
0.7254 \\
-0.0631 \\
0.7147 \\
-0.2050 \\
-0.1241 \\
1.4897 \\
1.4900 \\
1.4172
\end{bmatrix}
\begin{bmatrix}
0.6715 & -0.1022 & -1.0891 & 1.4193 & 0.8404 \\
-1.2075 & -0.2414 & 0.0326 & 0.2916 & -0.8880 \\
0.7172 & 0.3192 & 0.5525 & 0.1978 & 0.1001 \\
1.6302 & 0.3129 & 1.1006 & 1.5877 & -0.5445 \\
0.4889 & -0.8649 & 1.5442 & -0.8045 & 0.3035 \\
1.0347 & -0.0301 & 0.0859 & 0.6966 & -0.6003 \\
0.7269 & -0.1649 & -1.4916 & 0.8351 & 0.4900 \\
-0.3034 & 0.6277 & -0.7423 & -0.2437 & 0.7394 \\
0.2939 & 1.0933 & -1.0616 & 0.2157 & 1.7119 \\
-0.7873 & 1.1093 & 2.3505 & -1.1658 & -0.1941 \\
0.8884 & -0.8637 & -0.6156 & -1.1480 & -2.1384 \\
-1.1471 & 0.0774 & 0.7481 & 0.1049 & -0.8396 \\
-1.0689 & -1.2141 & -0.1924 & 0.7223 & 1.3546 \\
-0.8095 & -1.1135 & 0.8886 & 2.5855 & -1.0722 \\
-2.9443 & -0.0068 & -0.7648 & -0.6669 & 0.9610 \\
1.4384 & 1.5326 & -1.4023 & 0.1873 & 0.1240 \\
0.3252 & -0.7697 & -1.4224 & -0.0825 & 1.4367 \\
-0.7549 & 0.3714 & 0.4882 & -1.9330 & -1.9609 \\
1.3703 & -0.2256 & -0.1774 & -0.4390 & -0.1977 \\
-1.7115 & 1.1174 & -0.1961 & -1.7947 & -1.2078
\end{bmatrix}
\]

We then extracted, for each \( N \), the first \( N \) rows of \( a \) and \( B \) to generate the matrices \( A \) and \( B \), respectively. In the case of matrix \( B \), we extract its first \( m \) columns as well. Finally, the initial guess for the matrix \( K \) is the zero matrix, with ones in the position \( k_{jj}, j = 1, \ldots, m \). We limit the function count to 10000.

For the case \( m = 1 \), we tried two different values for the desired value of \( \mu \): \( \mu_1 = 1.5 \) and \( \mu = 0.5 \). Our results for \( m = 1 \) are presented in Tables 2-3 and we observe that when chose other values for the desired \( \mu_1 \), e.g. \( \mu_1 = 2 \), the algorithm does not provide us with a good solution. We did observe, however, that the results were better for this value of \( \mu_1 \) with increasing \( N \). Nevertheless, values closer to the initial value of \( \mu_1 \) yield better results. This is in agreement with the fact that there might not always be a solution to this problem, and conditions for existence of solutions are an object of current work.

| \( N \) | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|
| Run time (s) | 6.69 | 34.71 | 73.13 | 158.9 | 325.8 | 680.7 | 1451.1 |
| Nr of iterations | 991 | 1711 | 1455 | 1257 | 1127 | 1019 | 932 |
| Function Count | 4324 | 10002 | 10000 | 10001 | 10008 | 10002 | 10003 |
| \( E_1 \) | 1.1775 | 0.8133 | 0.8221 | 0.8854 | 0.6911 | 0.6310 | 0.6128 |
| \( E_2 \) | 0.0079 | 0.0940 | 0.0203 | 0.0373 | 0.0088 | 0.0030 | 0.0270 |

Table 2: Results of the numerical method for the solution of the general case with \( m = 1 \) and when we chose to change \( \mu = 1 = 1.5 \).

Another interesting test for our method is the case when we are willing to have more than one control to change only one eigenvalue. We tested our algorithm for \( N = 10 \), and \( m = 2, 3, 4, 5 \), and \( \mu_1 = 1.5, 0.5 \).

We observe that, in general, if we allow for the use of more controls then we obtain a better solution. Furthermore, the method performs better for the ‘worst case’ for \( \mu_1 \) (\( \mu_1 = 1.5 \)). This means that we can expect to be able to obtain
The first and second entries as µ our initial guess for solutions we would not be able to obtain for Table 4: Results of the numerical method for the two chosen values for µ.

Table 3: Results of the numerical method for the solution of the general case with m = 1 and when we chose to change µ = 1 = 0.5.

Table 4: Results of the numerical method for the two chosen values for µ1 and when we allow for the use of more controls.

solutions we would not be able to obtain for m = 1 (e.g., we were able to obtain µ1 = 2, which was not possible for our initial guess for K and m = 1, with E1 = 9.9 × 10−14 and E2 < 10−16, and using m = 3.

We then chose µ1 = 0.5 (which yielded the best results for m = 1, see tables 2-3), and for the case m = 2 we altered the first and second entries as µ1 = 0.5, µ2 = 4.

Table 5: Results of the numerical method for the solution of the general case with m = 2.

We did the same for m = 3, 4, 5 with µ1 = 0.5, µ2 = 4, µ3 = −5, µ1 = 0.5, µ2 = 4, µ3 = −5, µ4 = 2 and µ1 = 0.5, µ2 = 4, µ3 = −5, µ4 = 2, µ5 = 1, respectively. Our results are presented in Tables 6-8 below.

We observe that for the case m = 5 there is no point in using N = 4, since there would not be 5 eigenvalues to
It appears that as we increase the number of eigenvalues to change (and/or the number of controls to use), the method benefits from larger values of $N$. However, increasing $N$ much further than 10 makes the convergence of the nonlinear solver much slower. The results obtained are, however, satisfactory for the application our work was motivated from.

The final test case we perform is to use less controls than the number of eigenvalues we wish to change. It is known in control theory that, depending on the properties of $A$ and $B$ it is even possible to shift the whole spectrum with only one scalar control (for example, when $A$ is an $N \times N$ Jordan block and $B = e_N$ - the $N$-th unit basis vector - by the Hautus criterion the pair $(A,B)$ is controllable, so that by the Pole Placement Theorem the eigenvalues of $A + BK$ can be arbitrarily assigned by choosing a suitable $K$). Therefore, we investigate if this is possible to do with our algorithm.

We choose the last case we presented, i.e., change the first 5 eigenvalues of $2A$ to the values prescribed before, and as an example we use $m = 3$ controls. Our results are presented in Table 9 below.

We obtain satisfactory results, and in fact the algorithm is faster, and performs less iterations and function evaluations for $m = 3$. Furthermore, the error, as measured by $E_1$ and $E_2$ is smaller when using only $m = 3$ controls.

"Table 6: Results of the numerical method for the solution of the general case with $m = 3$."

| $N$ | 4   | 5   | 6   | 7   | 8   | 9   | 10  |
|-----|-----|-----|-----|-----|-----|-----|-----|
| Run time (s) | 0.12 | 0.33 | 0.77 | 1.79 | 4.01 | 6.98 | 16.33 |
| Nr of iterations | 7   | 8   | 8   | 8   | 8   | 8   | 6   |
| Function Count | 108 | 146 | 171 | 198 | 225 | 196 | 217 |
| $E_1$ | $2.4 \times 10^{-10}$ | $5.4 \times 10^{-15}$ | $5.5 \times 10^{-15}$ | $3.1 \times 10^{-15}$ | $1.1 \times 10^{-13}$ | $1.5 \times 10^{-10}$ | $2.6 \times 10^{-10}$ |
| $E_2$ | $2.3 \times 10^{-10}$ | $8.9 \times 10^{-16}$ | $4.4 \times 10^{-16}$ | $4.4 \times 10^{-16}$ | $1.1 \times 10^{-15}$ | $2.8 \times 10^{-13}$ | $5.3 \times 10^{-14}$ |

"Table 7: Results of the numerical method for the solution of the general case with $m = 4$."

| $N$ | 4   | 5   | 6   | 7   | 8   | 9   | 10  |
|-----|-----|-----|-----|-----|-----|-----|-----|
| Run time (s) | 0.20 | 0.56 | 1.22 | 3.17 | 8.07 | 22.08 | 48.52 |
| Nr of iterations | 6   | 8   | 7   | 8   | 9   | 11  | 11  |
| Function Count | 119 | 189 | 200 | 261 | 330 | 453 | 492 |
| $E_1$ | $1.2184$ | $0.4195$ | $0.8030$ | $0.6798$ | $0.5069$ | $0.4607$ | $0.5697$ |
| $E_2$ | $1.2011$ | $0.2772$ | $0.7804$ | $0.6190$ | $0.4754$ | $0.2296$ | $0.0406$ |

"Table 8: Results of the numerical method for the solution of the general case with $m = 5$."

| $N$ | 5   | 6   | 7   | 8   | 9   | 10  |
|-----|-----|-----|-----|-----|-----|-----|
| Run time (s) | 0.69 | 1.91 | 4.02 | 10.59 | 30.05 | 67.21 |
| Nr of iterations | 6   | 7   | 7   | 7   | 9   | 9   |
| Function Count | 182 | 248 | 252 | 328 | 460 | 510 |
| $E_1$ | $0.3558$ | $1.1378$ | $1.1811$ | $0.6859$ | $0.9958$ | $0.1111$ |
| $E_2$ | $0.2050$ | $1.0730$ | $0.0384$ | $0.6240$ | $0.4110$ | $0.0095$ |
provide updated eigenvalues. We shall examine these and related issues in future studies.

matrix in Ding & Zhou (2007) and it would be interesting to check if our algorithm may be simplified in this case to case presenting slower convergence.

sKS example, where we wish to change the already stable eigenvalues as little as possible. Numerical experiments suggest that solutions exist for diagonal matrices $A$ and $m$, as long as these matrices are symmetric (see Corollary 3.3).

Another interesting question is related to the initial guess for the matrix $K$. Here, we chose the initial guess to be $[I_m \ 0]$, where $I_m$ is the $m \times m$ identity matrix and 0 stands for the zero matrix of size $m \times (N - m)$ motivated by the sKS example, where we wish to change the already stable eigenvalues as little as possible. Numerical experiments started with the zero matrix, or a matrix whose entries are all equal to one produced similar results, with the second case presenting slower convergence.

It would also be interesting to establish the connection with similar work done on perturbations to the Google matrix in Ding & Zhou (2007) and it would be interesting to check if our algorithm may be simplified in this case to provide updated eigenvalues. We shall examine these and related issues in future studies.

| $N$  | 5    | 6    | 7    | 8    | 9    | 10   |
|------|------|------|------|------|------|------|
| Run time (s) | 0.31 | 1.03 | 1.53 | 3.65 | 9.15 | 18.15 |
| Nr of iterations | 6    | 6    | 6    | 7    | 8    | 7    |
| Function Count | 114  | 175  | 154  | 200  | 252  | 248  |
| $E_1$  | $1.3 \times 10^{-12}$ | 0.6477 | $8.4 \times 10^{-15}$ | $1.4 \times 10^{-12}$ | $9.9 \times 10^{-11}$ | $1.2 \times 10^{-9}$ |
| $E_2$  | $7.3 \times 10^{-13}$ | 0.6093 | $3.3 \times 10^{-16}$ | $1.2 \times 10^{-15}$ | $7.7 \times 10^{-14}$ | $5.8 \times 10^{-13}$ |

Table 9: Results of the numerical method for the solution of the general case, but when we wish to change 5 eigenvalues with $m = 3$ controls.

5 Analysis, Conclusions and Further Work

We have outlined a method for finding a matrix $K$ such that we can prescribe eigenvalues to the matrix $D = 2A + BK + K'B'$, when we are given an $N \times N$ diagonal matrix $A$ and an $N \times m$ full matrix $B$. The motivation was a particular application in linear control theory of sPDEs. We demonstrate the good performance of the algorithm by showing its good convergence in the two metrics we used to measure the error in each iteration. We note that we were only interested in the case when the matrix $A$ is diagonal, but the method presented here is expected to work for full matrices $A$, as long as these matrices are symmetric (see Corollary 3.3).

We present the results of the application of our method for two different examples, for various values of $N$ and small values of $m$, which were sufficient for the application we considered. In the sKS example, we observe good convergence of the algorithm in both measures we consider, at least for small $N$. This is explained through the fact that the matrix $B$ has a regular structure and $A$ has repeated eigenvalues. We observe that the algorithm converges faster when the distance between the diagonal entries of $2A$ and the required eigenvalues is small. There may be difficulties in wanting to make a perturbation that is too large, and this idea still needs to be quantified.

We focus mostly on using the same number of controls as eigenvalues to change due to the application that motivated us: here we are already using the same number of controls as unstable modes (i.e., the eigenvalues we need to change) for the equation (1.3) and so there is no necessity to use less controls here. However, we have also tested the algorithm for cases when we use more and less controls than the number of eigenvalues we wish to prescribe and again found good convergence, which is in agreement with other pole placement algorithms.

The proposed algorithm performance improves with increased $N$ in the sense that the running time decreases with increasing $N$ for the randomly generated matrices $A$ and $B$. The maximum running time for the sKS equation example is large, but for the purposes of the application this was sufficient, as it was a small addition to the already long computational time.

Finally, there are a number of questions related to the analysis presented here. Existence and uniqueness of solutions of the system of nonlinear equations is an interesting and not straightforward problem to study. In fact, it is easy to establish conditions on the desired eigenvalues and the entries of the matrix $A$ for existence of solutions in the case $N = 2, m = 1$, but this problem becomes considerably harder as we increase $N$ or $m$. Our numerical experiments suggest that solutions exist for diagonal matrices $A$ and $m$ at least as large as the number of eigenvalues we wish to change.

Another interesting question is related to the initial guess for the matrix $K$. Here, we chose the initial guess to be $[I_m \ 0]$, where $I_m$ is the $m \times m$ identity matrix and 0 stands for the zero matrix of size $m \times (N - m)$ motivated by the sKS example, where we wish to change the already stable eigenvalues as little as possible. Numerical experiments started with the zero matrix, or a matrix whose entries are all equal to one produced similar results, with the second case presenting slower convergence.

It would also be interesting to establish the connection with similar work done on perturbations to the Google matrix in Ding & Zhou (2007) and it would be interesting to check if our algorithm may be simplified in this case to provide updated eigenvalues. We shall examine these and related issues in future studies.
Acknowledgements

We are grateful to the anonymous referees for insightful comments and valuable suggestions and to Profs. S. Kalliadasis, D. T. Papageorgiou and G. A. Pavliotis for numerous stimulating discussions, insightful comments and encouragement. This research was supported by Imperial College London through a Roth Ph.D. studentship and the EPSRC via Grants No. EP/L025159/1, EP/L020564/1, EP/L027186/1 and EP/J009636/1.

References

BARABASI, A.-L. & STANLEY, H.E.(1995) Fractal Concepts in Surface Growth. Cambridge University Press.

BARTELS, R.H. & STEWART, G.W.(1972) Solution of the matrix equation $AX + XB = C$ [F4]. Commun. ACM, 15(9), 820-826.

BRADEN, H.W.(1998) The equations $A^T X \pm X^T A = B$ SIAM J. Matrix Anal. Appl., 20(2), 295–302.

DE TERÁN, F., DOPICO, F.M., GUILLERY, N., MONTEALEGRE, D. & REYES, N.(2013) The solution of the equation $AX + X^* B = 0$ Linear Algebra Appl., 438, 2817–2860.

DING, J. & ZHOU, A.(2007) Eigenvalues of rank-one updated matrices with some applications. Appl. Math. Lett., 20(12), 1223-1226.

GOMES, S.N., KALLIADASIS, S., PAPAGEORGIOU, D.T., PAVLIOTIS, G.A. & PRADAS, M.(2016) Controlling roughening processes in the stochastic Kuramoto-Sivashinsky equation. Physica D. 348, 33–43.

HAGER, W.W.(1989) Updating the inverse of a matrix. SIAM Rev. 31(2), 221-239.

HU, G., LOU, Y. & CHRISTOFIDES, P.D.(2008) Dynamic output feedback covariance control of stochastic dissipative partial differential equations. Chem. Eng. Sci., 63, 4531–4542.

JIMENEZ, J.C.(2015) Simplified formulas for the mean and variance of linear stochastic differential equations. Appl. Math. Lett., 49,12–19.

KAUTSKY, J., NICHOLS, N. K. & VAN DOOREN, P.(1985) Robust pole assignment in linear state feedback. Int. J. Control, 41(5),1129–1155.

LEVENBERG, K.(1944) A method for the solution of certain nonlinear problems in least squares. Q. Appl. Math., 164-168.

LOU, Y. & CHRISTOFIDES, P.D.(1985) Feedback control of surface roughness in sputtering processes using the stochastic Kuramoto-Sivashinsky equation. Comput. Chem. Eng., 29,741–759.

MAHONY, R., HELMKE, U. & MOORE, J.(1993) Pole placement algorithms for symmetric realisations. Decision and Control, Proceedings of the 32nd IEEE Conference, pp. 1355-1358. An algorithm for least-squares estimation of nonlinear parameters. J. Soc. Indust. Appl. Math. 11(2), 431-441.

MARQUARDT, D.W.(1963) An algorithm for least-squares estimation of nonlinear parameters. J. Soc. Indust. Appl. Math. 11(2), 431-441.

PRADAS, M., PAVLIOTIS, G.A., KALLIADASIS, S., PAPAGEORGIOU, D.T. & TSELUIKO, D.(2012) Additive noise effects in active nonlinear spatially extended systems. Eur. J. Appl. Math. 23, 563-591.

PRADAS, M., TSELUIKO, D., KALLIADASIS, S., PAPAGEORGIOU, D.T. & PAVLIOTIS, G.A.(2011) Noise induced state transitions, intermittency, and universality in the noisy Kuramoto-Sivashinsky equation. Phys. Rev. Lett. 106, 060602.
PRESS, W.H., TEUKOLSKY, S.A., VETTERLING, W.T. & FLANNERY, B.P.(1972) Numerical Recipes 3rd Edition: The art of scientific computing. *Cambridge University Press.*

SHERMAN, J. & MORRISON, W.J.(1950) Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *Ann. Math. Stat.* **21**(1), 124-127.

SYLVESTER, J.J.(1884) Sur l’équation en matrices $px = xq$. *CR Acad. Sci. Paris,* **99**, 67-71 and 115-116.

WOODBURY, M.A.(1950) Inverting modified matrices. *Memorandum report 42,* 106.

YANG, K. & ORSI, R.(2006) Generalized pole placement via static output feedback: a methodology based on projections. *Automatica* **42**, 2143–2150.

ZABCZYK, J.(1992) Mathematical Control Theory: An Introduction. *Birkhauser.*