Methods for Lyapunov equations

This chapter is about numerical methods for a particular type of equation expressed as a matrix equality.

Definition 4.0.1. *Consider two square matrices $A, W \in \mathbb{R}^{n\times n}$. The problem to find a square matrix $X \in \mathbb{R}^{n\times n}$ such that*

$$AX + XA^T = W \tag{4.1}$$

is called the Lyapunov equation.

The Lyapunov equation is the most common problem in the class of problems called *matrix equations*. Other examples of matrix equations: Sylvester equation, Stein equation, Riccati equation.

Different algorithms are suitable for different situations, depending on the properties of A and W, and we work out two algorithms. For dense A, the *Bartels-Stewart algorithm* is one of the most efficient approaches. The Bartels-Stewart algorithm is derived in Section 4.2. For sparse and large-scale A and if W is of low-rank, Krylov-type methods may be more efficient (Section 4.3).

Some applications are given in Section 4.4 and in the exercises.

Traditionally driven by certain problems in system and control, the Lyapunov equation now appears in very large number of fields. The developments and improvements of numerical methods have (and continues to be) recognized as important in numerical linear algebra.

A naive approach

Equation (4.1) is a linear system of equations, expressed in a somewhat unusual form. We will now see that (4.1) can be reformulated as a linear system of equations in standard form, by using techniques called vectorization and Kronecker products. If $B \in \mathbb{R}^{n\times m}$ with columns $(b_1, \ldots, b_m) = B$, the vectorization operation is defined as the stacking of the columns into a vector,

$$\text{vec}(B) := \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} \in \mathbb{R}^{nm}.$$

For two matrices $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{j\times k}$, the Kronecker product ($\otimes$) is defined as

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1m}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \cdots & a_{nm}B \end{bmatrix} \in \mathbb{R}^{nj\times mk}.$$

In the field of systems and control the equation (4.1) is sometimes called the *continuous-time* Lyapunov equation, for disambiguation with a different matrix equation called the *discrete-time* Lyapunov equation. Many of the algorithms we present here can be adapted for discrete-time Lyapunov equations.

In matlab, the vec-operation can be computed with `b=B(:)` and the inverse operation can be computed with `B=reshape(b,n,length(b)/n)`. The Kronecker product is implemented in `kron(A,B)`.

With this notation we can derive a very useful identity. For matrices A, B, X of matching size, we have

$$\operatorname{vec}(AXB) = (B^T \otimes A)\operatorname{vec}(X) \tag{4.2}$$

By vectorizing the Lyapunov equation (4.1) we have

$$\begin{aligned} \operatorname{vec}(AX + XA^T) &= \operatorname{vec}(W) \\ (I \otimes A + A \otimes I)\operatorname{vec}(X) &= \operatorname{vec}(W) \end{aligned} \tag{4.3}$$

Apply (4.2) twice, once with $A = I$ and once with $B = I$.

The equation (4.3) is a linear system of equations on the standard matrix-times-vector form. The connection between (4.1) and (4.3) is useful, mostly for theoretical purposes. For instance, we can easily characterize the existance and uniqueness of a solution.

Theorem 4.1.1 (Existance and uniqueness of solution)**.** *Let λ_i, $i = 1, \ldots, n$ be the eigenvalues of A.*

(i) The equation (4.1) has a unique solution $X \in \mathbb{R}^{n\times n}$ if and only if

$$\lambda_i \neq -\lambda_j, \text{ for all } i, j = 1, \ldots, n.$$

(ii) In particular, if A is strictly stable (that is $\lambda_i < 0$, for all $i = 1, \ldots, n$), then (4.1) has a unique solution.

The eigenvalues of a strictly stable matrix have negative real part.

Proof. This is an exercise. □

A naive computational approach based on (4.3)

It is tempting to approach the problem of solving the Lyapunov equation (4.1) by applying a generic method for linear systems of equations $Bz = w$ to (4.3),

$$\operatorname{vec}(X) = (I \otimes A + A \otimes I)^{-1}\operatorname{vec}(W). \tag{4.4}$$

Suppose for simplicity that we have a numerical method that can solve a linear system of equations with N with $\mathcal{O}(N^3)$ operations (such as a primitive implementation of Gaussian elimination). Since (4.3) is a linear system with $N = n^2$, the computational complexity of such an approach is

$$t_{\text{naive}}(n) = \mathcal{O}(n^6). \tag{4.5}$$

A generic method for linear systems applied to (4.3), will have high computational complexity.

Some gains in complexity are feasable by using more advanced versions of Gaussian elimination to solve (4.3), or exploiting sparsity in $I \otimes A + A \otimes I$. These improved variants will not be computationally competitive for large n in comparison to the algorithms in the following sections.

Although this approach is not competitive for large n, it can be useful for small n, and (as we shall see below) as a component in a numerical method for large n.

Bartels-Stewart algorithm

From previous lectures we know that there are efficient algorithms that can compute the Schur decomposition of a matrix. It turns out that this can be used as a precomputation such that we obtain a triangular Lyapunov equation. The resulting equation can be efficiently solved in direct way with a finite number of operations, by using a type of substitution.

The Bartels-Stewart algorithm, initially presented for slightly more general problems in [1] and is one of the leading methods for dense Lyapunov equations. It is implemented in matlab in the command `lyap`.

Recall that all real matrices have a real Schur decomposition: There exists matrices Q and T such that

$$A = QTQ^T,$$

For efficiency reasons, we here work with the *real* Schur form, where in contrast to the *complex* Schur form, the complex triangular matrix is replaced by a real block triangular matrix with blocks of size one or two as in (4.6)

where Q is an orthogonal matrix and $T \in \mathbb{R}^{n\times n}$ a block-triangular matrix, where

$$T = \begin{bmatrix} T_{11} & \cdots & T_{1,r} \\ & \ddots & \vdots \\ & & T_{r,r} \end{bmatrix} \tag{4.6}$$

and $T_{jj} \in \mathbb{R}^{n_j\times n_j}$, $n_j \in \{1,2\}$, $j = 1,\ldots,r$ and $\sum_{j=1}^{r} n_j = n$.

We multiply the Lyapunov equation (4.1) from the right and left with Q and Q^T respectively,

$$\begin{aligned} Q^TWQ &= Q^TAXQ + Q^TXAQ = && \text{(4.7a)} \\ &= Q^TAQQ^TXQ + Q^TXQQ^TAQ = && \text{(4.7b)} \\ &= TY + YT^T && \text{(4.7c)} \end{aligned}$$

Use $QQ^T = I$.

where $Y = Q^TXQ$. We introduce matrices and corresponding blocks such that

$$Q^TWQ = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}, Y = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix}, \quad T = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}, \tag{4.8}$$

where the blocks are such that $Z_{22}, C_{22}, T_{rr} = R_{22} \in \mathbb{R}^{n_r\times n_r}$ (the size of the last block of T). This triangularized problem can now be solved with (what we call) backward substitution, similar to backward substitution in Gaussian elimination. By separating the four blocks in the equation (4.7), we have four equations

$$\begin{aligned} C_{11} &= R_{11}Z_{11} + R_{12}Z_{21} + Z_{11}R_{11}^T + Z_{12}R_{12}^T && \text{(4.9a)} \\ C_{12} &= R_{11}Z_{12} + R_{12}Z_{22} + Z_{12}R_{22}^T && \text{(4.9b)} \\ C_{21} &= R_{22}Z_{21} + Z_{21}R_{11}^T + Z_{22}R_{12}^T && \text{(4.9c)} \\ C_{22} &= R_{22}Z_{22} + Z_{22}R_{22}^T. && \text{(4.9d)} \end{aligned}$$

General idea: Solve (4.9b)-(4.9d) explicitly. Insert the solution into (4.9a) and repeat the process for the smaller equation which is a Lyapunov equation with unknown $Z_{11} \in \mathbb{R}^{(n-n_r)\times(n-n_r)}$.

Due to the choice of block sizes, the last equation of size $n_r \times n_r$, which can be solved explicitly since $n_r \in \{1,2\}$:

- If $n_r = 1$, the equation (4.9d) is scalar and we obviously have

$$Z_{22} = \frac{C_{22}}{2R_{22}}. \tag{4.10}$$

- If $n_r = 2$ we can still solve (4.9d) cheaply since it is a 2×2 Lyapunov equation. For instance, we can use the naive approach (4.4), i.e.,

$$\operatorname{vec}(Z_{22}) = (I \otimes R_{22} + R_{22} \otimes I)^{-1} \operatorname{vec}(C_{22}). \tag{4.11}$$

The eigenvalues of R_{22} are eigenvalues of A. Therefore the small Lyapunov equation (4.9d) has a unique solution if (4.1) has a unique solution.

Insert the now known matrix Z_{22} into (4.9b) and transposed (4.9c), yields

$$\tilde{C}_{12} := C_{12} - R_{12}Z_{22} = R_{11}Z_{12} + Z_{12}R_{22}^T \tag{4.12a}$$
$$\tilde{C}_{21} := C_{21}^T - R_{12}Z_{22}^T = R_{11}Z_{21}^T + Z_{21}^T R_{22}^T. \tag{4.12b}$$

The equations (4.12) have a particular structure that can be used to directly compute the solutions Z_{12} and Z_{21}. An explicit procedure for the construction of the solution (4.12) is given by the following result.

Lemma 4.2.1. *Consider two matrices C, D partitioned in blocks of size $n_1 \times n_p$, $n_2 \times n_p$, ..., $n_{p-1} \times n_p$ according to*

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_{p-1} \end{bmatrix} \in \mathbb{R}^{N \times n_r}, \quad D = \begin{bmatrix} D_1 \\ \vdots \\ D_{p-1} \end{bmatrix} \in \mathbb{R}^{N \times n_p}$$

where $C_j, D_j \in \mathbb{R}^{n_j \times n_p}$ and $N = \sum_{j=1}^{p-1} n_j$. Let $U \in \mathbb{R}^{n \times n}$ be a block triangular matrix partitioned as X and D. For any $R_{22} \in \mathbb{R}^{n_p \times n_p}$, we that if X satisfies the equation

$$D = UX + XR_{22}^T, \tag{4.13}$$

then X_j, $p-1, p-2, \ldots, 1$ satisfy

$$U_{jj}X_j + X_j R_{22}^T = \tilde{D}_j \tag{4.14}$$

where $\tilde{D}_j := D_j - \sum_{i=j+1}^{p-1} U_{ji}X_i$.

The block triangular matrix U is

$$U = \begin{bmatrix} U_{11} & \cdots & U_{1,p-1} \\ & \ddots & \vdots \\ & & U_{p-1,p-1} \end{bmatrix}$$

where $U_{i,j} \in \mathbb{R}^{n_i \times n_j}$.

Similar to the approach we used to compute Z_{22}, X_j can expressed and computed explicitly from a small linear system

$$\operatorname{vec}(X_j) = (I \otimes T_{jj} + R_{22} \otimes I)^{-1} \operatorname{vec}(\tilde{W}_j) \tag{4.15}$$

By solving (4.15) for $j = p-1, \ldots, 1$, for both equations (4.12)a and (4.12)b we obtain solutions Z_{12} and Z_{21}. Insertion of (the now known solutions) Z_{12}, Z_{21} and Z_{22} into (4.9a) gives a new Lyapunov equation of size $n - n_p$ and the process can be repeated for the smaller matrix.

Compute the real Schur decompositon `[Q,T]=schur(A)` and establish $n_1, \ldots, n_r$, $T_{11}, T_{1r}, \ldots, T_{rr}$ with partitioning according to (4.6)
Set $C = Q^T W Q$.
Set $m = n$.
for k=$r,\ldots,1$ **do**
  Set $m = m - n_k$
  Partition the matrix C with $C_{11}, C_{12}, C_{21}, C_{22}$ according to (4.8), with $C_{22} \in \mathbb{R}^{n_k \times n_k}$.
  Set $R_{22} = T_{kk}$, and

$$R_{11} = \begin{bmatrix} T_{11} & \cdots & T_{1,k-1} \\ & \ddots & \vdots \\ & & T_{k-1,k-1} \end{bmatrix}, \quad R_{12} = \begin{bmatrix} T_{1,k} \\ \vdots \\ T_{k-1,k} \end{bmatrix}$$

  Solve (4.9d) for $Z_{22} \in \mathbb{R}^{n_k \times n_k}$ using (4.11) or (4.10).
  Compute $\tilde{C}_{12}$, $\tilde{C}_{21}$ using (4.12)a and (4.12)b
  Solve (4.12)a and (4.12)b for $Z_{12} \in \mathbb{R}^{m \times n_k}$ and $Z_{21} \in \mathbb{R}^{n_k \times m}$ using equation (4.14) and (4.15) with $p = k$.
  Store $Y(1:m, m+(1:n_k)) = Z_{12}$
  Store $Y(m+(1:n_k), 1:m) = Z_{21}$
  Store $Y(m+(1:n_k), m+(1:nk)) = Z_{22}$
  Set $C := C_{11} - R_{12}Z_{21} - Z_{12}R_{12}^T$.
end
Return solution $X = QYQ^T$.

Algorithm 1: The Bartels-Stewart algorithm for the Lyapunov equation

Complexity of Bartels-Stewart algorithm

Earlier in the course we learned that the QR-algorithm could be tuned to perform well. In practice the total complexity is usually roughly estimated by $\mathcal{O}(n^3)$. Since the other parts of the algorithm (see exercise) has complexity $\mathcal{O}(n^3)$, the total complexity of Bartels-Stewart algorithm is

$$t_{\text{Bartels–Stewart}}(n) = \mathcal{O}(n^3).$$

This is a tremendous improvement in comparison to the naive approach (4.5), estimated by $\mathcal{O}(n^6)$.

Low-rank methods for large and sparse problems

Although the Bartels-Stewart is both efficient and robust for large dense problems, there is very little room to use sparsity and other structures. We now discuss a class of approaches which are suitable

for certain Lyapunov equations, under the condition that $W = ww^T$ and A is a stable matrix (and large and sparse). Under these conditions, we now show that the solution can be accurately approximated with a low-rank matrix.

Low-rank approximability has been extended to other conditions on the matrices, e.g., when W has a small rank.

Low-rank approximability of solution

In other parts of the course we have shown that if A is a stable matrix, we can express the solution to the Lyapunov equation with an integral of matrix exponentials

$$X = -\int_0^\infty \exp(tA)W\exp(tA^T),. \tag{4.16}$$

The low-rank property of X is now illustrated by constructing an explicit approximation of X by applying Gauss-quadrature to the indefinite integral of the matrix in (4.16). More precisely, we now construct an approximation $\tilde{X} \approx X$ by setting

$$\tilde{X} = -\sum_{i=1}^m \omega_i \exp(t_i A)W\exp(t_i A^T), \tag{4.17}$$

where $(\omega_1, t_1), \ldots, (\omega_m, t_m)$ are appropriately selected points and weights.

Recall now that a Gauss-quadrature gives a way to approximate integrals by a weighted sum. For appropriately selected evaluation points $t_1, \ldots, t_m$ and weights $\omega_1, \ldots, \omega_m$ we can approximate

$$\int_0^\infty f(t)\,dt \approx \sum_{i=1}^m \omega_i f(t_i)$$

We have the following important observation if $W = ww^T$: Every term in (4.17) is a rank-one matrix. Hence, a quadrature formula with m points results in an explicit way to construct an approximation $\tilde{X}$ which has

$$\operatorname{rank}(\tilde{X}) \leq m. \tag{4.18}$$

The approximation $\tilde{X}$ is only accurate if the Gauss-quadrature approximation is accurate. We know from the theory of Gauss-quadrature that weights and points can be selected to yield accurate approximations for scalar-valued Gauss-quadrature. The following result show how the Gauss-quadrature error for the matrix function can be characterized.

For simplicity assume A is symmetric such that $A = V\Lambda V^T$ where V is an orthogonal matrix and $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. We now get an explicit expression for the error

$$\begin{aligned} &X - \tilde{X} \\ &= V\left(\int_0^\infty \exp(t\Lambda)\tilde{w}\tilde{w}^T\exp(t\Lambda)\,dt - \sum_{i=1}^m \omega_i \exp(t_i\Lambda)\tilde{w}\tilde{w}^T\exp(t_i\Lambda)\right)V^T \\ &= VZ_mV^T \end{aligned}$$

where $\tilde{w} = V^T w$. Moreover, the individual elements in Z_m can be bounded by

$$|(Z_m)_{j,k}| \leq |\int_0^\infty e^{t(\lambda_j+\lambda_k)} - \sum_1^m \omega_i e^{t(\lambda_j+\lambda_k)}||\tilde{w}_j\tilde{w}_k| \tag{4.19}$$

We immediately identify a condition which leads to a small approximation error. The approximation error for $\tilde{X}$ is small if we simultaneously approximate integrals

$$\int_0^\infty e^{t(\lambda_j+\lambda_k)}\,dt$$

for all j and k. We summarize this in a theorem.

Theorem 4.3.1. *Suppose A is symmetric and stable and suppuse W is symmetric and has rank one. Let $(t_1,\omega_1),\ldots,(t_m,\omega_m)$ be a quadrature method and define*

$$\varepsilon_m := \max_{j,k} \left| \int_0^\infty e^{t(\lambda_j+\lambda_k)}\,dt - \sum_1^m \omega_i e^{t(\lambda_j+\lambda_k)} \right|.$$

Then, there exists a matrix $\tilde{X}$ such that

$$\begin{aligned} \operatorname{rank}(\tilde{X}) &\leq m \\ \|X-\tilde{X}\|_{\max} &\leq \varepsilon_m \|w\|^2 \end{aligned}$$

A specific choice of a quadrature formulas leads to specific expressions for the error. One specific choice (used for instance by Grasedyck) leads to an error estimate of the type

$$\|X-\tilde{X}\| \leq \alpha e^{-\sqrt{m}}.$$

Galerkin approach to the Lyapunov equation

We saw above that *there exists* a low-rank approximation of X under certain conditions. Although we were able to derive explicit formulas for $\tilde{X}$ in the previous section, these formulas are typically not used in the computation. Rather than evaluating the formulas for the low-rank approximation, we say that we directly try to compute a low-rank solution by trying to determine $U \in \mathbb{R}^{n\times m}$ and $P \in \mathbb{R}^{m\times m}$ such that

$$\tilde{X} = UPU^T,$$

where U is an orthogonal matrix.

Hence, we wish to have,

$$A\tilde{X} + \tilde{X}A^T = AUPU^T + UPU^TA^T \approx W.$$

The most common procedure to find (and compute) an approximation directly is by the application of a Galerkin condition. More precisely, given U we compute P by imposing the Galerkin condition and simplifying

$$\begin{aligned} U^T\left(A\tilde{X}+\tilde{X}A^T\right)U &= U^TWU \\ U^TA\tilde{X}U + U\tilde{X}A^TU &= U^TWU \\ U^TAUPU^TU + U^TUPU^TA^TU &= U^TWU \\ U^TAUP + PU^TA^TU &= U^TWU \\ \tilde{A}P + P\tilde{A}^T &= \tilde{W} \end{aligned}$$

where $\tilde{A} = U^T AU$. This is a small dense Lyapunov equation which can be solved directly, for instance with the Bartels-Stewart method.

The matrix U corresponds to a basis of a search subspace (or projection space). The success of the approach highly depends on the choice of U and many different search subspaces have been proposed in the literature.

In the paper by V. Simoncini [5] it is proposed to use a so called extended Krylov subspace. It illustrates the exploitation of low-rank in methods for matrix equations.

Applications

The research and developments on numerical methods for Lyapunov equations was initially driven by applications in the field of systems and control. There are now applications appearing in many completely different fields. Some specific applications are listed below.

The so-called infinite-time gramian of continuous time linear dynamical system is the solution to a Lyapunov equation. They can be used

- to study stability, controllability and observability;
- to compute the $\mathcal{H}_2$-norm which is important in for instance controller design;
- to solve PDEs on tensorized domains;
- to compute bounds and characterisations of transient effects of ODEs;
- to design an optimal control; and
- to compute a reduced order model with balanced truncation.

The Lyapunov equation also appears as a part in numerical methods for stability radius, and in the study of robust stability of large sparse time-delay systems.

A discretization of Helmholtz equation $\Delta u = f$ on a square domain with homogeneous boundary conditions using finite difference with uniform grid, leads to a linear system of equations $(D_{xx} \otimes I + I \otimes D_{xx})u = b$. This is a vectorized Lyapunov equation. It can consequently be solved efficiently with, e.g., Bartels-Stewart method. A Lyapunov equation approach can be useful for more advanced discretizations of more complicated PDEs, by using a Lyapunov equation solver in the preconditioning.

There are also applications in numerical methods for matrix functions, tensor approximation problems (used for instance in computational methods for the electronic Schrödinger equation), ...

Lyapunov equations are used in various situations at KTH. They appear in the courses *SF2842 - Geometric Control Theory*, *FEL3500 - Introduction to Model Order Reduction (Dept. Automatic control)* and *EL2620 - Nonlinear Control*. It is also used in research, e.g., the research of the author of these lecture notes and researchers at KTH Mechanics department use Lyapunov equations to carry out approximations of flow `ftp://www2.mech.kth.se/pub/shervin/review.pdf`.

Other matrix equations and other methods

There are a considerable number of matrix equations appearing in various fields, which can often be considered extensions of the Lypuanov equation, e.g., the discrete-time Lyapunov equation, Sylvester equation generalized Lyapunov equation [6], or Ricatti equation, delay Lyapunov equations or time-dependent Lyapunov equations (e.g. by KTH-researcher Henrik Sandberg [4]). There are many differende numerical methods apart from those mentioned here, e.g., Hammerlings method [3] (which attractive in combination with model order reduction and was later improved by Sorensen). There are methods which consider the Lyapunov equation as optimization problem in a differential geometric setting, more precisely optimization over Riemannian manifolds (in particular [7] which has been recognized with various prizes and award).

For further reading see the survey paper

`http://www.dm.unibo.it/~simoncin/matrixeq.pdf`

and the wiki for model order reduction

`http://morwiki.mpi-magdeburg.mpg.de`

Exercises

1. Implement the naive approach as well as the Bartels-Stewart method based on the skeleton code on the course web page. Compare the methods by using simulations for random matrices. Illustrate (with figure) the dependence of the computation time on n. What is the observed complexity?

   Skeleton code for the Bartels-Stewart method: `http://www.math.kth.se/~eliasj/NLA/lyap0_skeleton.m`

2. Prove (4.2).

3. Show that
$$\text{vec}(uv^T) = v \otimes u.$$
for any $u, v \in \mathbb{R}^n$.

4. Consider the problem
$$A = \begin{bmatrix} 0 & a \\ 1 & 0 \end{bmatrix}, \; W = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}.$$
For which values of a does the Lyapunov equation have a unique solution?

5. Prove the following identity: Any matrices A, B, C, D of compatible size satisfy
$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD).$$

   Hint for a short proof: Let $\{A_i\}, \{B_j\}, \{C_k\}, \{D_\ell\}$ form bases of the matrix spaces associated with A, B, C, D. A function $f(A, B, C, D)$ which is linear in all parameters is $f(A, B, C, D) \equiv 0$, if $f(A_i, B_j, C_k, D_\ell) = 0$ for all combinations of basis vectors.

6. Prove Theorem 4.1.1.
   Hint: Use previous exercise to first show that if v_j, v_i are two eigenvectors of A, then $v_j \otimes v_i$ is an eigenvector of $I \otimes A + A \otimes I$.

7. Prove Lemma 4.2.1.

8. Derive the computational complexity of computing $X_1, \ldots, X_{p-1}$ with Lemma 4.2.1. You may here assume that $n_1 = \cdots = n_p = 2$.

9. Show that if A and W are symmetric, then a solution X to (4.1) is symmetric, if (4.1) has a unique solution.

10. Derive a more efficient algorithm for the case when A and W are symmetric, by first showing that (4.12)a and (4.12)b have the same solution. Illustrate that it works and that it is more efficient using the code from exercise 1. (More difficult optional: Also use that T in the Schur decomposition of A is a diagonal matrix.)

11. Consider the following dynamical system
$$\begin{aligned} \dot{x}(t) &= Ax(t) + bu(t) \\ y(t) &= c^T x(t), \end{aligned}$$
where $b, c \in \mathbb{R}^n$. Suppose A is stable, i.e., $\lambda(A) \in \mathbb{C}_-$.

   In systems and control, this is called a single-input single-output (SISO) dynamical system.

a) The observability gramian of this system is defined as

$$P_c(t) := \int_0^t \exp(A^T t) c c^T \exp(At)\, dt$$

and the observability gramian

$$P_o(t) := \int_0^t \exp(At) b b^T \exp(A^T t)\, dt.$$

Show that the limits

$$\begin{aligned} P_o(t) &\to P_{o,\infty} \in \mathbb{R}^{n\times n} \text{ as } t \to \infty \\ P_c(t) &\to P_{c,\infty} \in \mathbb{R}^{n\times n} \text{ as } t \to \infty \end{aligned}$$

exist and satisfy certain Lyapunov equations.

b) The so-called $\mathcal{H}_2$-norm of the dynamical system is defined as

$$\|\Sigma\|_{\mathcal{H}_2} := \sqrt{\frac{1}{2\pi}\int_{-\infty}^{\infty} |c^T(i\omega I - A)^{-1} b|^2\, ds}.$$

Show that

$$\|\Sigma\|_{\mathcal{H}_2}^2 = \frac{1}{2\pi}\int_0^\infty y(t)^2\, dt$$

The $\mathcal{H}_2$-norm is a measure of sensitivity of the system, and can be used in controller synthesis when designing control systems which should be insensitive to additive (stochastic) white-noise.

c) Show that the $\mathcal{H}_2$ norm can be computed from either controllability or observability gramian by showing

$$\|\Sigma\|_{\mathcal{H}_2}^2 = c^T P_{o,\infty} c \tag{4.20a}$$
$$= b^T P_{c,\infty} b \tag{4.20b}$$

In the matlab control toolbox, in particular in the function `norm(sys,2)`, a Lyapunov equation is solved with Bartels-Stewart algorithm. The $\mathcal{H}_2$-norm is subsequently computed directly from the relation (4.20).

12. Consider the partial differential-equation

$$\begin{aligned} \Delta u + g(x,y)u &= f(x,y) \text{ for } (x,y) \in \Omega \\ u(x,y) &= 0 \text{ for } (x,y) \in \partial\Omega, \end{aligned}$$

where $\Omega = [0,1]\times[0,1]$.

(a) Derive the (second order) finite-difference discretization for the grid $x_i = hi$, $i = 1,\ldots,m$, $y_j = hj$, $j = 1,\ldots,m$ and $h = 1/(m+1)$. Derive matrices $D_{xx}, G, F \in \mathbb{R}^{n\times n}$ such that the discretization can be expressed as

$$D_{xx}U + UD_{xx} + G \circ U = F, \tag{4.21}$$

Here $\circ$ denotes the Hadamard product, also known as the direct or element-wise product, in matlab `.*`

for $U_{i,j} \approx u(x_i, y_j)$.

(b) Derive explicit expression for the eigenvalues of $I \otimes D_{xx} + D_{xx} \otimes I$ in the limit $m \to \infty$, and show that $I \otimes D_{xx} + D_{xx} \otimes I$ is nonsingular in the limit.
(Optional: The eigenvalues have closed forms also for finite m. Compute these and show that $I \otimes D_{xx} + D_{xx} \otimes I$ is non-singular for any m.)

(c) Let $g(x,y) = \alpha\sqrt{(x-\frac{1}{2})^2 + (y-\frac{1}{2})^2}$ and $f(x,y) = |x-y|$. Solve the problem in the following ways, and report computation time as a function of system size:

- For $\alpha = 0$, solve the sparse linear system corresponding to the vectorization of (4.21) with the matlab command \
- Compare with solving (4.21) with the matlab command `lyap` also with $\alpha = 0$.
- For $\alpha = 1$, solve the sparse linear system corresponding to the vectorization of (4.21) with the matlab command \.
- For $\alpha = 1$, apply GMRES to solve the sparse linear system corresponding to the vectorization of (4.21) (use matlab command `gmres`)
- For $\alpha = 1$, apply GMRES to solve, by using `lyap` as a left preconditioner.
- For $\alpha = 0.1$, apply GMRES to solve, by using `lyap` as a left preconditioner.

Use only sparse matrices and do not store any matrices as full matrices. Use `spdiags` and see `help spdiags`.

(d) Explain the performance of the preconditioned GMRES in the previous sub-problem using the theory developed earlier in the course, preferably in terms explicit bounds involving the analytic expressions for the eigenvalues (in the limit or for finite n).

Might be helpful: From theory of FrÃl'chet derivates of matrix functions we know that for sufficiently small $\|E\|$, we have

$$(A-E)^{-1} = A^{-1} - A^{-1}EA^{-1} + \mathcal{O}(\|E\|^2).$$

This exercise has many solutions.

(e) Suppose all elements of G are zero except $G_{m/4,m/2} = 1/h$ and assume $m \in 4\mathbb{Z}$. Solve the equation efficiently using `lyap` (or your implementation of Bartels-Stewart). This problem can and should here be solved without using gmres or similar. Compare with a naive approach.

Hint: Consider the problem as rank-one modification and read about the Sherman-Morrison-Woodbury formula.

This choice of G corresponds to the regularization of dirac impulse in the point $(x,y) = (\frac{1}{4}, \frac{1}{2})$.

13. This exercise is about K-PIK described in [5]. You will not need to read all the details of the paper to answer the questions.

(a) Suppose $V_k \in \mathbb{R}^{n\times k}$ is an orthogonal matrix and that $B \in \text{span}(V_1)$ where $B \in \mathbb{R}^n$. Prove that the approximation $X_k = V_k Y_k V_k^T$ gives a residual $R_k = AX_k + X_k A^T + BB^T$ which satisfies

$$R_k = V_{k+1}\underline{H}_k Y_k V_k^T + V_k Y_k \underline{H}_k^T V_{k+1}^T + V_k E E^T V_k.$$

under the conditions used to derive [5, Equation (2.3)]. What is $\underline{H}_k$ and E? What assumed about V_k?

(b) Suppose V_k in (a) is known. Derive a formula for Y_k, which gives $X = X_k$ if the solution to the Lyapunov equation can be expressed as $X = V_k Y_k V_k^T$.

(c) The approximation $X_k = V_k Y_k V_k^T$ where $V_k \in \mathbb{R}^{n\times k}$, $k \ll n$ corresponds to assuming that the exact solution X can be well approximated with a matrix with low rank. Fortunately, it turns out that the solution to the Lyapunov equation can be well approximated with a low-rank matrix in many cases. This can be precisely described theoretically by characterizing the singular values of X in terms of s and the eigenvalues of A. For instance, postive definite matrices and negative definite matrices result in a faster singular value decay than indefinite matrices. We will in this exercise only verify the decay of the singular values of the solution to Lyapunov numerically. Let $A = A_0 - tI$ be and $B = b \in \mathbb{R}^n$ be generated as follows.

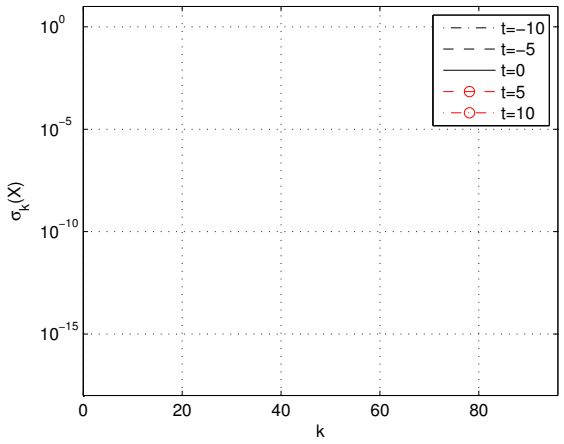
Recall the low-rank approximation property of singular values: For any matrix $C \in \mathbb{R}^{n\times n}$,

$$\min_{\substack{C_k \in \mathbb{R}^{n\times n} \\ \text{rank}(C_k)=k}} \|C - C_k\|_2 = \sigma_{k+1}(C)$$

where σ_k is the kth singular value, ordered by decreasing magnitude. For more information about the decay of singular values of the solution to Lyapunov equation see references in [2].

```
>> rand('seed',0); nx=5;
>> A0=-gallery('wathen',nx,nx);
>> n=length(A0); A=A0-t*speye(n,n);
>> b=eye(n,1);
```

Generate the following figure, where X is the solution to the Lyapunov equation (computed with `lyap` or `lyap0`) and $\sigma_k(X)$ can be computed with `svd(X)`.



(d) Run K-PIK on the problem in (c) for the different values of t, including the values of t in the figure. Interpret the observations in (c) and relate to how well K-PIK works, i.e., if it computes an accurate solution and how many iterations (`length(er2)`) are required to reach a specified tolerance. Increase the size of the problem by increasing `nx`. What is the largest problem you can solve with `lyap` and `kpik` with five seconds of computation time (and not running out of memory).

The matlab code for the K-PIK algorithm is available on `http://www.dm.unibo.it/~simoncin/software.html` The code is for a more general case. You can set `E=LE=speye(n,n)`

14. This exercise concerns a primitive (non-optimized) variant of `kpik`.
Let the matrix $A \in \mathbb{R}^{n\times n}$ be generated by

```
julia> function spectral_abscissa(A);
          ev,xv=eigs(A,which=:LR);
          I=indmax(real(ev));
          return real(ev[I]);
       end
julia> n=100; srand(0);  # reset random seed
julia> A=sprandn(n,n,0.1);
julia> s=spectral_abscissa(A);
julia> alpha=1;
julia> A=A-speye(size(A,1))*(alpha+s);
julia> new_s=spectral_abscissa(A)
-1.0000000000000373
julia> b=randn(n);
```

A former student of SF2524 has made kpik and several other matrix equation methods available for julia: https://github.com/garrettthomaskth/LargeMatrixEquations.jl

(a) Plot the singular value decay of the solution to the Lyapunov equation with $W = bb^T$ for different α-values, and experimentially determine a sufficient α such that there exists a rank 5 approximation of the Lyapunov equation which which is of order of magnitude 10^{-10} from the exact solution. In other words, determine α such that there exists $\tilde{X}$ with $\text{rank}(\tilde{X}) \leq 5$ and

$$\|X - \tilde{X}\| \leq 10^{-10}.$$

For this exercise you may use `X=lyap(full(A),b*b')`.

(b) `kpik` corresponds to a projection on the subspace

$$\text{span}(q_1, \ldots, q_m) = \mathcal{K}_{m+1}(A, b) \quad (4.22a)$$
$$\text{span}(g_1, \ldots, g_m) = \mathcal{K}_{m+1}(A^{-1}, b) \quad (4.22b)$$

By computing the matrices $Q = [q_1, \ldots, q_m]$ and $G = [g_1, \ldots, g_m]$ separately (with code you have computed previously in the course) we can compute an orthogonal basis of $\text{span}(u_1, \ldots, u_{2m-1})$ with the commands

Constructing two separate Krylov subspaces and subsequently merging them with an call to qr() as we do in this exercise, is in general not efficient, but sufficient to obtain understanding of the method in this case.

```
julia> P,H=arnoldi...
julia> G,H=arnoldi..
julia> U,R=qr(hcat(P,G));
julia> U=U[:,find(abs(diag(R)) .> 100*eps())]; # remove duplicate b vector
```

Construct an approximate solution $\tilde{X} = UPU^T$ by using the Galerkin approach. Plot the approximation error $\|\tilde{X} - X\|$ as a function m for $\alpha = 1, 2, \ldots$.

(c) Compare the CPU-time your approach in (b) with the `lyap` for different values of n:

```
julia> @time lyap(full(A),b*b');
 0.022112 seconds (57 allocations: 749.859 KB)
```

For which parameter values α is the low-rank approach more efficient? Can you beat lyap? (In a fully optimized version of `kpik` this is possible, but for this equivalent but primitive variant, it depends on implementation details and your computer)

Project suggestions

Graduate level projects related to Lyapunov equations:

- Alternating Direction Implicit (ADI)
- Bartels-Stewart algorithm for the Sylvester equation
- Balancing and balanced trunctation: Use the Lyapunov equation for model order reduction
- Efficient algorithm for the discrete-time Lyapunov equation
- Complex Bartels-Stewart algorithm, derive algorithm based on the complex Schur decomposition..

References

[1] R. Bartels and G. W. Stewart. Solution of the matrix equation $AX + XB = C$. *Comm A.C.M.*, 15(9):820–826, 1972.

[2] L. Grubišić and D. Kressner. On the eigenvalue decay of solutions to operator Lyapunov equations. *Syst. Control Lett.*, 73:42–47, 2014.

[3] S. J Hammarling. Numerical solution of the stable, non-negative definite lyapunov equation lyapunov equation. 2(3):303–323, 1982.

[4] H. Sandberg. An extension to balanced truncation with application to structured model reduction. *IEEE Trans. Autom. Control*, 55, 2010.

[5] V. Simoncini. A new iterative method for solving large-scale Lyapunov matrix equations. *SIAM J. Sci. Comput.*, 29(3):1268–1288, 2007.

[6] T. Stykel. Stability and inertia theorems for generalized Lyapunov equations. *Linear Algebra Appl.*, 355:297–314, 2002.

[7] B. Vandereycken and S. Vandewalle. A Riemannian optimization approach for computing low-rank solutions of Lyapunov equations. *SIAM J. Matrix Anal. Appl.*, 31(5):2553–2579, 2010.