SF3580
HW 4

Anna Broms & Fredrik Fryklund

January 25, 2019

Task 1

We implement the naive approach for solving the Lyapunov equation and compare it to the Bartels-Stewart method on random matrices. The dependence of the simulation time on n is illustrated in Figure 2. Theoretically, the complexity is as bad as $\mathcal{O}(n^6)$ for the naive approach and $\mathcal{O}(n^3)$ for the improved algorithm. However, the simulation time is in this test smaller due to the use of improved linear solvers in Matlab. Tests were done with the Matlab `timeit` command and the system was solved 10 times for each matrix size, such that an avarage computation time could be determined. From our investigations, we conclude that the theoretical complexities should be seen as upper bounds for the complexity. The measured complexity is $\mathcal{O}(n^{4.30})$ and $\mathcal{O}(n^{1.80})$ respectively (if fitted to a straight line in the loglog-plot) and $\mathcal{O}(n^{4.38})$ and $\mathcal{O}(n^{1.76})$ if the `cputime` command is used for the measurements instead.
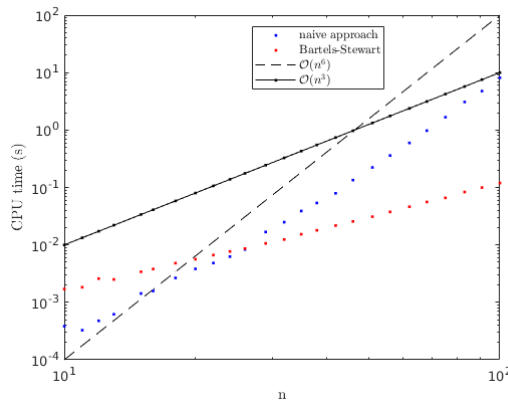

Figure 1: Measured CPU-time for solving the Lyapunov equation on random matrices of size n using a naive approach compared to the Bartels-Stewart algorithm.

Task 3

We show that $\text{vec}(\mathbf{u}\mathbf{v}^T) = \mathbf{v} \otimes \mathbf{u}$, where $\mathbf{u}$ and $\mathbf{v}$ are two vectors of length n. Starting with the left hand side, we have

$$uv^T = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \begin{bmatrix} v_1, & v_2, & \ldots, & v_n \end{bmatrix} = \begin{bmatrix} u_1v_1, & u_1v_2, & \ldots, & u_1v_n \\ u_2v_1 & u_2v_2, & \ldots, & u_2v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_nv_1 & u_nv_2 & \ldots & u_nv_n \end{bmatrix}. \tag{1}$$
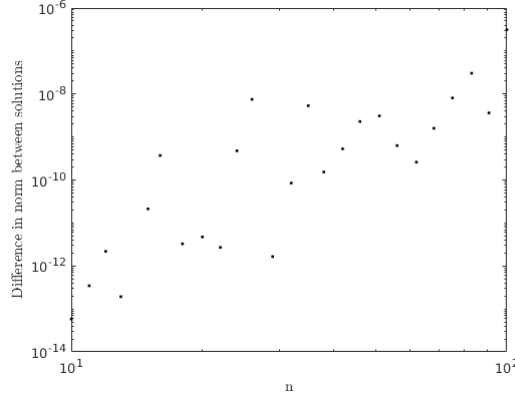
Figure 2: Difference in norm between the naive approach and the Bartels-Stewart algorithm for solving the Lyapunov equation on random matrices of size n.

Thus,

$$\text{vec}(\mathbf{u}\mathbf{v}^T) = \begin{bmatrix} u_1v_1, \ldots, u_nv_1, u_1v_2, \ldots, u_nv_2, \ldots, u_1v_n, \ldots, u_nv_n \end{bmatrix}. \tag{2}$$

The right hand side expression can be written as

$$\mathbf{v} \otimes \mathbf{u} = \begin{bmatrix} v_1\mathbf{u} \\ v_2\mathbf{u} \\ \vdots \\ v_n\mathbf{u} \end{bmatrix}, \tag{3}$$

which is the same as the expression for the left hand side in (2).

Task 4

We consider

$$A = \begin{bmatrix} 0 & a \\ 1 & 0, \end{bmatrix} W = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \tag{4}$$

and determine for which values of a that the Lyapunov equation has a unique solution.

The answer is given using theorem 4.1.1 from the lecture notes. The eigenvalues of the real matrix A is $\lambda = \pm\sqrt{a}$. The theorem states that the Lyapunov equation has a unique solution if and only if $\lambda_1 \neq -\lambda_2$. The only possibility for this to hold is if $a = 0$.

Task 5

Let $A \in \mathbb{R}^{m_A \times k}$, $B \in \mathbb{R}^{m_B \times p}$, $C \in \mathbb{R}^{k \times n_C}$ and $D \in \mathbb{R}^{p \times n_D}$. Then the matrices A, B, C and D are of compatible size such that they satisfy $(A \otimes B)(C \otimes D)$ and $(AC) \otimes (BD)$. We show that $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$ by first computing $(A \otimes B)$ and $(C \otimes D)$:

$$(A \otimes B) = \begin{bmatrix} a_{11}B & \cdots & a_{1k}B \\ \vdots & \ddots & \vdots \\ a_{m_A1}B & \cdots & a_{m_Ak}B \end{bmatrix} \in \mathbb{R}^{(m_A m_B) \times (kp)}, \quad (C \otimes D) = \begin{bmatrix} c_{11}D & \cdots & c_{1n_C}D \\ \vdots & \ddots & \vdots \\ c_{k1}D & \cdots & c_{kn_C}D \end{bmatrix} \in \mathbb{R}^{(kp) \times (n_C n_D)}.$$

2

We now have

$$(A \otimes B)(C \otimes D) = \begin{bmatrix} a_{11}B & \cdots & a_{1k}B \\ \vdots & \ddots & \vdots \\ a_{m_A 1}B & \cdots & a_{m_A k}B \end{bmatrix} \begin{bmatrix} c_{11}D & \cdots & c_{1n_C}D \\ \vdots & \ddots & \vdots \\ c_{k1}D & \cdots & c_{kn_C}D \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=1}^{k} a_{1i}c_{i1}BD & \cdots & \sum_{i=1}^{k} a_{1i}c_{in_C}BD \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{k} a_{m_A i}c_{i1}BD & \cdots & \sum_{i=1}^{k} a_{m_A i}c_{in_C}BD \end{bmatrix}$$

where the matrix product CD is well defined. Now consider the right hand side of (??)

$$(AC) \otimes (BD) = \begin{bmatrix} \sum_{i=1}^{k} a_{1i}c_{i1} & \cdots & \sum_{i=1}^{k} a_{1i}c_{in_C} \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{k} a_{m_A i}c_{i1} & \cdots & \sum_{i=1}^{k} a_{m_A i}c_{in_C} \end{bmatrix} \otimes (BD)$$

$$= \begin{bmatrix} \sum_{i=1}^{k} a_{1i}c_{i1}BD & \cdots & \sum_{i=1}^{k} a_{1i}c_{in_C}BD \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{k} a_{m_A i}c_{i1}BD & \cdots & \sum_{i=1}^{k} a_{m_A i}c_{in_C}BD \end{bmatrix} = (A \otimes B)(C \otimes D).$$

Thus we have $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$.

Task 12

Consider the PDE

$$\Delta u + g(x,y)u = f(x,y), \text{ for } x,y \in \Omega, \qquad (5)$$
$$u(x,y) = 0, \text{ for } x,y \in \partial\Omega,$$

where Ω is the unit square.

(a)

Derive the 2nd order finite-difference discretization for the grid $x_i = hi, i = 1,\ldots,m, y_j = hj, j = 1,\ldots,m$ and $h = 1/(m+1)$. Also, derive matrices such that the discretizatioin can be expressed as

$$D_{xx}U + UD_{xx} + G \circ U = F, \qquad (6)$$

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

Using a 2nd order approximation of the second derivatives

$$\frac{\partial^2 f}{\partial x^2} = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \qquad (7)$$

and similarly for $\frac{\partial^2 f}{\partial y^2}$, we can approximate the laplacian by the five point stencil

$$\Delta u_{i,j} \approx \frac{u_{i-1,j} - 4u_{i,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}}{h^2}, \qquad (8)$$

for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, m$. From the boundary condition, we have

$$u_{0,j} = u_{m+1,j} = u_{i,0} = u_{m+1,0} = 0. \tag{9}$$

The structure for the Laplace operator alone is thus the $m^2 \times m^2$ matrix

$$\begin{bmatrix} T & I & 0 & \ldots & 0 \\ I & T & I & \ldots & \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & I & T \end{bmatrix} \tag{10}$$

with T being the $m \times m$ matrix with -4 on the diagonal, 1 on the sub- and super-diagonals and zero otherwise. The entire system for the PDE can be written on the form $Au = b$, where A has the same structure as the matrix in (10), but where the diagonal is given by matrices $B_1, B_2, \ldots, B_m$, where $B_j = T + h^2 \cdot GG_j$, with GG_j having values only on the main diagonal taking the values $g(x_i, y_j)$, $i = 1, \ldots, m$. The right hand side vector b is given by

$$b = h^2 \begin{bmatrix} f(x_i, y_1) \\ f(x_i, y_2) \\ \vdots \\ f(x_i, y_m) \end{bmatrix}. \tag{11}$$

Observe that this is the vectorized form of the equation (6), where D_{xx} is a $m \times m$ matrix with -2 on the main diagonal and -1 on the sub- and superdiagonals, $G = h^2 \cdot g(x_i, y_j)$ and $F = h^2 \cdot f(x_i, y_j)$.

(b)

Derive explicit expressions for the eigenvalues of $I \otimes D_{xx} + D_{xx} \otimes I$ in the limit $m \to \infty$ and show that the matrix $I \otimes D_{xx} + D_{xx} \otimes I$ is non-singular in the limit.

First, we identify that the matrix $I \otimes D_{xx} + D_{xx} \otimes I$ is the finite difference approximation of the laplace operator. In the limit $m \to \infty$, the approximation approaches the continuous laplace operator. Therefore, we seek eigenvalues

$$\Delta u = \lambda u. \tag{12}$$

Writing u as $u(x, y) = X(x)Y(y)$, we get the equation

$$X'' + Y'' = \lambda XY \Leftrightarrow \frac{X''}{X} + \frac{Y''}{Y} = \lambda, \tag{13}$$

enforcing that for some constants α and β,

$$X'' = \alpha X \text{ and } Y'' = \beta Y. \tag{14}$$

Solving these equations along with the homogeneous boundary conditions, we obtain that

$$\lambda = (k\pi)^2 + (l\pi)^2, \quad k, l = 1, 2, 3, \ldots \tag{15}$$

The equations for $X(x)$ and $Y(y)$ have no nontrivial solutions for $\lambda = 0$ and thus it can be concluded that the matrix is non-singular.
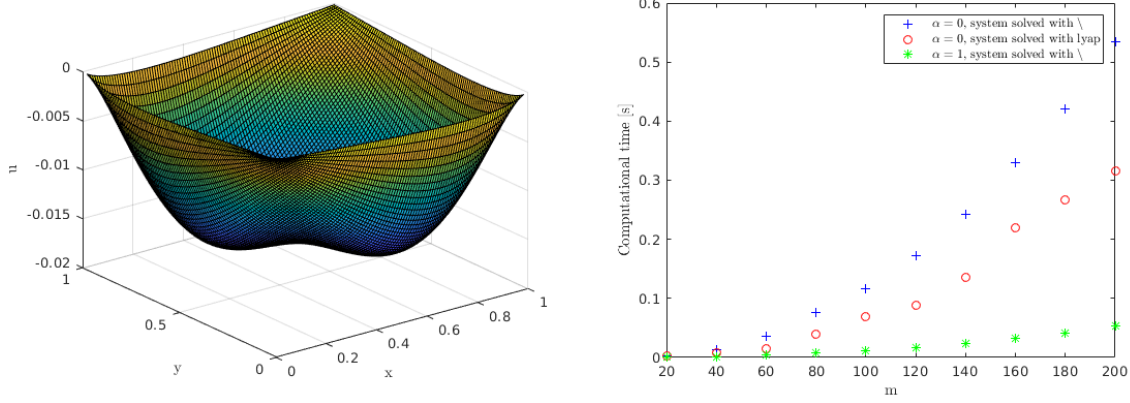

(c)

Letting

$$g(x, y) = \alpha \sqrt{(x - \frac{1}{2})^2 + (y - \frac{1}{2})^2} \tag{16}$$

and $f(x, y) = |x - y|$, we do the following (comparing solution techniques and timing results):

1. Let $\alpha = 0$ and solve the sparse linear system $Au = b$ using \.

2. We compare the above with solving the equation using the matlab command `lyap`.

3. Let $\alpha = 1$ and solve with \

(a) Solution task 12: 1. and 2. computed with $m = 100$.

(b) Timeing results for tasks 12: 1-3.

Figure 3: Tasks 1-3

Computing the difference between the solutions in task 12 1) and 2) with $m = 100$ yiels $\|U_1 - U_2\| \approx 1.37 \times 10^{-12}$ and the average time when solving the system 10 times for different matrix sizes and choices for α is found in Figure 3b. It can therefore be concluded that it is significantly faster to solve the matrix equation using `lyap` instead of solving the linear system using \ and also that the system faster can be solved with \ if $\alpha = 1$ than if $\alpha = 0$.

4. Again, let $\alpha = 1$, but use instead `gmres` to solve the system.

   We conclude that gmres fails to converge to desired tolerance within a reasonable number of iterations. Setting $m = 100$, the relative residual after RESTART*MAXIT= 20×10 iterations is 0.0886.

5. Using $\alpha = 1$ and `gmres`, use `lyap` as a left pre-conditioner.

   A left preconditioner is a matrix M modyfing the system $Au = b$ such that the new system to solve is $M^{-1}Au = M^{-1}b$. A function computing the action of an inverted preconditioner matrix applied to a vector z can be formed using `lyap`$(T, -Z * h^2)$, where $Z =$`reshape(z,m,m)` and where the result from the `lyap` command is reshaped to a vector of size $m^2 \times 1$. Solving the system using $m = 100$, gmres and the described left-preconditioner, we can compute the norm $\|U_{\text{backslash}} - U_{\text{gmres with precond}}\| \approx 1.50 \times 10^{-9}$.

6. Using $\alpha = 0.1$ and `gmres`, use `lyap` as a left pre-conditioner.

   We compare the computational time for an average of 10 solves using \ and $\alpha = 1$, pre-conditioned `gmres` and $\alpha = 1$ and pre-conditioned `gmres` and $\alpha = 0.1$. The result is visualised in Figure 4. It can be concluded that the pre-conditioned `gmres` performes better in terms of computation time for $\alpha = 0.1$ than for $\alpha = 1$. It should also be noted that solving with backslash is faster than using gmres, which might be due to how we have chosen to construct the preconditioner-function.

(d)

Explain the performance in (c) of the the preconditioned gmres. Consider $A - E$ to be the matrix A obtained by choosing $\alpha = 0$. Then, from the hint in the lecture notes,

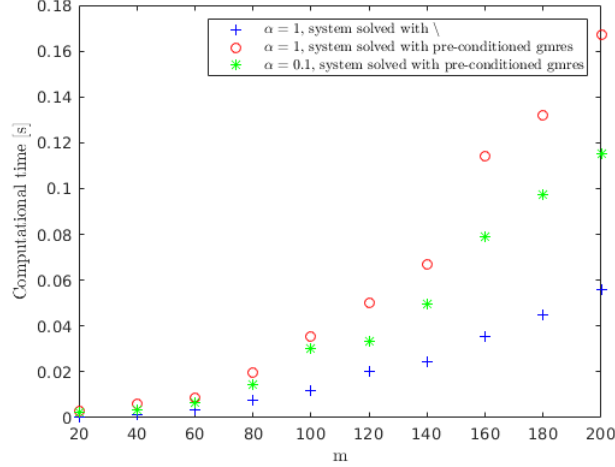$$(A - E)^{-1} = A^{-1} - A^{-1}EA^{-1} + \mathcal{O}(\|E\|^2), \tag{17}$$

5

Figure 4: Timing results comparing the preconditioned gmres for different α and also comparing to results using \.

for sufficiently small $\|E\|$. The matrix E is diagonal, having only the values $h^2 \cdot g(x_i, y_j)$ on the diagonal ordered columnwise, and therefore, $\|E\|_2 = h^2 \cdot \max(g(x_i, y_2)) < h^2 \cdot \alpha \sqrt{2}/2$. Thus, using the solution from the lyap command (with $\alpha = 0$) as a pre-conditioner, the inverse of the preconditioner $M = A - E$ is very close to being the inverse of the matrix A where e.g. $\alpha = 0.1$. The consequence is clearly that gmres converges in few iterations as the product $(A - E)^{-1} A \approx I$.

(e)

Suppose all elements of the matrix G are zero except $G_{m/4, m/2} = 1/h$, where $m \in 4\mathbb{Z}$. Solve the equation efficiently by using the `lyap` command.

We consider the problem as a rank-one modification and use the Sherman-Morrison-Woodbury formula, stating that if A is an invertible square matrix and u, v are column vectors of the same size, then $A + uv^T$ is invertible iff $1 + v^T A^{-1} u \neq 0$. The inverse is given by

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1} uv^T A^{-1}}{1 + v^T A^{-1} u}. \tag{18}$$

We want to solve a large and sparse linear system of the form

$$(A + \tilde{G})x = b, \tag{19}$$

where $\tilde{G}$ is the sparse matrix putting the values of the matrix G on the diagal columnwise. Using that $\tilde{G} = uv^T$, we obtain

$$(A + \tilde{G})x = b \quad \Leftrightarrow \quad (A + uv^T)x = b \quad \Leftrightarrow x = (A - uv^T)^{-1}b. \tag{20}$$

Now, applying the Sherman-Morrison-Woodbury formula, (20) can be rewritten as

$$x = (A - uv^T)^{-1}b = A^{-1}b - \frac{A^{-1} uv^T A^{-1} b}{1 + v^T A^{-1} u}. \tag{21}$$

We identify that for the specified matrix G, $u = e_{(m/2-1)m+m/4}$ and $v = (1/h) \cdot e_{(m/2-1)m+m/4}$. The problem thus boils down to solving a lyapunov equation twice with different right hand sides and then vectorize the results. In pseudo-code, we compute

$$\begin{aligned} &\alpha = \texttt{lyap}(T, -F); \\ &\alpha = \alpha(:); \\ &\beta = \texttt{lyap}(T, -e_{(m/2-1)m+m/4}); \\ &\beta = \beta(:). \end{aligned} \tag{22}$$

6

Then,

$$x = (A - uv^T)^{-1}b = \alpha - \frac{\beta(v^T\alpha)}{1 + v^T\beta}. \tag{23}$$

We compare with a naive approach solving (19) using \. Using $m = 100$, the 2-norm difference between the solutions is 1.37×10^{-12}.

Task 13

(a)

Consider the approximate solution $X_k = V_k Y_k V_k^T$ for the Lyapunov equation

$$AX + XA^T + BB^T = 0, \tag{24}$$

where the columns of V_k spans the subspace K_k of $\mathbb{R}^n$. The columns are orthonormal. Furthermore, let $B = V_K E$ for some E. Introduce the residual

$$R_k = AX_k + XA_k^T + BB^T. \tag{25}$$

Consider a sequence of approximation spaces $K_k \subset K_{k+1}$ that satisfies $AK_k \subset K_{k+1}$. If V_k spans K_k, then K_{k+1} contains AV_k, which is spanned by the columns of V_{k+1}. Or in other words: AV_k can be expressed in K_{k+1} as $V_{k+1}\underline{H}_k$, which can be interpreted as an Arnoldi iteration. We can now rewrite the residual R_k as

$$R_k = V_k\underline{H}_k V_{k+1}^T X_k + XV_{k+1}^T \underline{H}_k^T V_k + V_k EE^T V_k^T. \tag{26}$$

(b)

Impose that the residual (25) is orthogonal to K, i.e.

$$V_k^T R_k V_k = 0 \Leftrightarrow V_k^T AV_k Y_k V_k^T V_k + V_k^T V_k Y_k V_k^T A_k^T V_k + EE^T = 0 \tag{27}$$

$$\Leftrightarrow V_k^T AV_k Y_k + Y_k V_k A^T V_k^T + EE^T = 0. \tag{28}$$

From (28) we have that

$$V_k^T AV_k Y_k + Y_k V_k A^T V_k^T + EE^T = 0,$$

and if $VYV^T = X$ is exact, then

$$V^T AVY + YV A^T V^T + EE^T = 0.$$

Introduce $H_k = V_k^T AV_k$. From the lecture notes we note that the approximate solutions Y_k and X_k are given by

$$Y_k = \int_0^\infty e^{tH_k} EE^T e^{tH_k^T} dt \Leftrightarrow V_k Y_k V_k^T = \int_0^\infty V_k e^{tH_k} EE^T e^{tH_k^T} V_k^T dt = X_k.$$

and analytical solution X satisfies

$$X = \int_0^\infty e^{tA} BB^T e^{tA^T} dt.$$

We know that $X = V_k Y_k V_k^T$, thus $H = V^T AV = V_k^T AV_k = H_k$ as well. With $B = V_k E$ we have

$$X = \int_0^\infty e^{tA} BB^T e^{tA^T} dt = \int_0^\infty e^{tV_k HV_k^T} V_k EE^T V_k^T e^{tV_k H^T V_k^T} dt = \int_0^\infty V_k e^{tH} V_k^T V_k EE^T V_k^T V_k e^{tH^T} V_k^T dt$$

$$= \int_0^\infty V_k e^{tH} EE^T e^{tH^T} V_k^T dt = X_k.$$

That $e^{tV_k H V_k^T} = V_k e^{tH} V_k^T$ is a simple consequence of V_k being orthonormal. By interpreting $e^{tV_k H V_k^T}$ via the Taylor series definition the matrices V_k and V_k^T will cancel out. We have

$$e^{tV_k H V_k^T} = \sum_{i=0}^{\infty} \frac{t^i}{i!} (V_k H V_k^T)^i = V_k \left(\sum_{i=0}^{\infty} \frac{t^i}{i!} H^i \right) V_k^T = V_k e^{tH} V_k^T.$$

(c)

The claim that the singular values of positive, or negative, definite matrices converge faster than for indefinite matrices is supported by the data plotted in Figure 5.
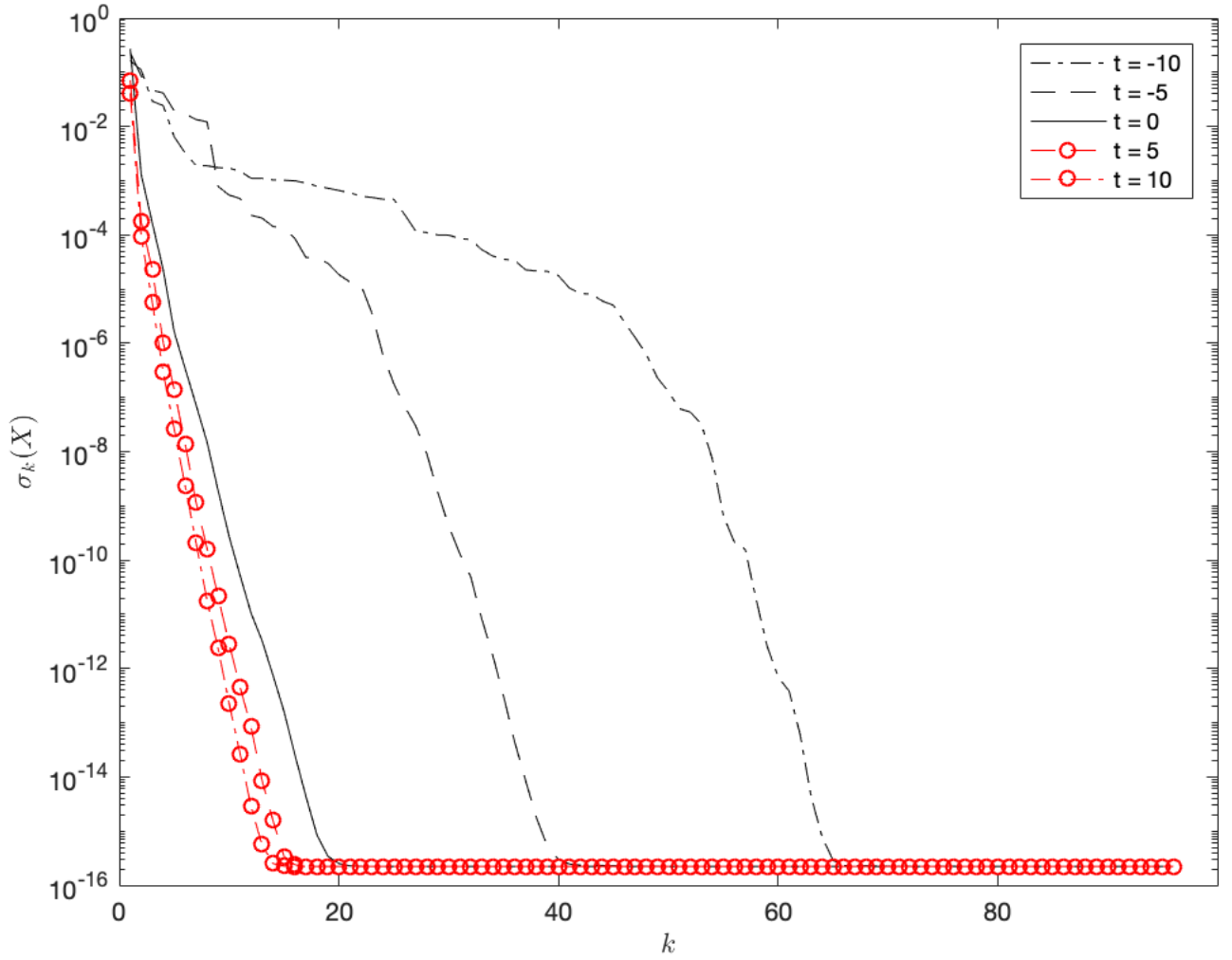

Figure 5: Task 13(c).

(d)

The timings are the mean of ten runs, with time measured with Matlab's `tic` and `toc`. The largest problem I can solve on my machine, withing 5 seconds, is $nx = 167$ with `k-pik` and $nx = 17$ for `lyap`,

8

for $t \in [0, 1, 10, 100, 1000]$. There is some small variation in nx for `k-pik` with different t, but no more than ± 2. For $nx = 17$ `lyap` gives an error about $1e - 14$ for the given range of t. For `k-pik` $t = 0$ and $t = 1$ the error are of magnitude $1e - 11$ and $1e - 12$, respectively. For the other values of t the error is $1e - 14$. The number of iterations decrease with t, however for $t = 0, 1, 10$ all solves with `lyap` takes ten iterations. For $t = 100$ and $t = 1000$ the number of iterations are 7 and 4, respectively.

The `k-pik` method can treat larger systems faster than `lyap`, and it is clear that the larger t, the smaller rank is required to approximate the solution X well. This is evident from the number of iterations, which corresponds to the dimension of the approximation space. We saw the same trend in 5, where a large t, i.e. a definite matrix, implied that few singular values are needed to represent the solution well. This relates to that a low rank approximation suffices to obtain a small error.