SF3580
HW 4

Anna Broms & Fredrik Fryklund

January 23, 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.

Fredrik: kan du frklara komplexiteten bttre?

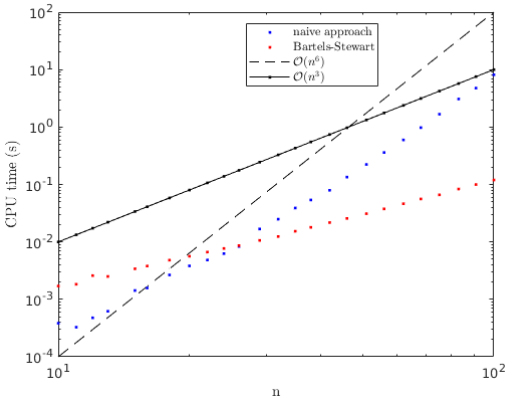


Figure 1: Measured CPU-time for solving the Lyapunov equation on random matrices of size n using a naive appoach compared to the Bartels-Stewart algorithm.

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

$$uv^T = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \begin{bmatrix} v_1, & v_2, & \dots, & v_n \end{bmatrix} = \begin{bmatrix} u_1v_1, & u_1v_2, & \dots, & u_1v_n \\ u_2v_1 & u_2v_2, & \dots, & u_2v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_nv_1 & u_nv_2 & \dots & u_nv_n \end{bmatrix}. \tag{1}$$

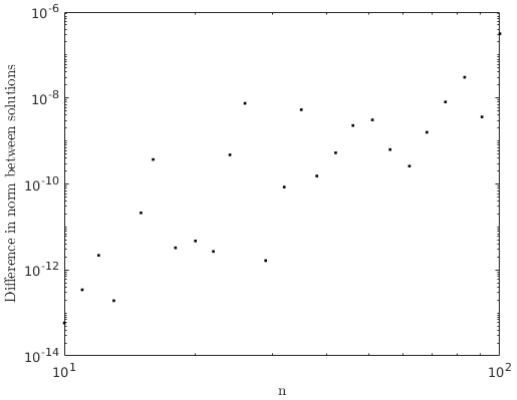


Figure 2: Difference in norm between the naive approach and the Bartels-Stewart algorithm for soving the Lyapunov equation on random matrices of size n.

Thus,

$$\text{vec}(\mathbf{u}\mathbf{v}^T) = \left[u_1v_1, \ldots, u_nv_1, u_1v_2, \ldots, u_nv_2, \ldots, u_1v_n, \ldots, u_nv_n\right]. \tag{2}$$

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 the left hand side of

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD) \tag{5}$$

is satisfied. This is shown by first computing $(A \otimes B)$ and $(C \otimes D)$:

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

We now have

$$(A \otimes B)(C \otimes D) = \begin{bmatrix} a_{11}B & \cdots & a_{1k}B \\ \vdots & \ddots & \vdots \\ a_{m_A 1}B & \cdots & a_{m_A k}B \end{bmatrix} \begin{bmatrix} c_{11}D & \cdots & c_{1n_C}D \\ \vdots & \ddots & \vdots \\ c_{k1}D & \cdots & c_{kn_C}D \end{bmatrix}$$
$$= \begin{bmatrix} \sum_k^{i=1} a_{1i}c_{i1}BD & \cdots & \sum_k^{i=1} a_{1i}c_{in_C}BD \\ \vdots & \ddots & \vdots \\ \sum_k^{i=1} a_{m_A i}c_{i1}BD & \cdots & \sum_k^{i=1} a_{m_A i}c_{in_C}BD \end{bmatrix}$$

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

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

Task 12

Consider the PDE

$$\begin{aligned} \Delta u + g(x,y)u &= f(x,y), \text{ for } x, y \in \Omega, \\ u(x,y) &= 0, \text{ for } x, y \in \partial\Omega, \end{aligned} \tag{6}$$

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

$$D_{xx}U + UD_{xx} + G \circ U = F, \tag{7}$$

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

Using a 2nd order approximation of the second derivatives

$$\frac{\partial^2 f}{\partial x^2} = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \tag{8}$$

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

$$\Delta u_{i,j} \approx \frac{u_{i-1,j} - 4u_{i,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}}{h^2}, \tag{9}$$

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

$$u_{0,j} = u_{m+1,j} = u_{i,0} = u_{m+1,0} = 0. \tag{10}$$

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

$$\begin{bmatrix} T & I & 0 & \dots & 0 \\ I & T & I & \dots & \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & I & T \end{bmatrix} \tag{11}$$

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 $B_1, B_2, \dots, B_m$, where $B_j = T + h^2 \cdot GG_j$, with GG_j having values only on the main diagonal taking the values $g(x_i, y_j)$, $i = 1, \dots, m$. The right hand side vector b is given by

$$b = h^2 \begin{bmatrix} f(x_i, y_1) \\ f(x_i, y_2) \\ \vdots \\ f(x_i, y_m) \end{bmatrix}. \tag{12}$$

Observe that this is the vecorized 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, $G = g(x_i, y_j)$ and $F = f(x_i, y_j)$.

(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

$$\Delta u = \lambda u. \tag{13}$$

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

$$X'' + Y'' = \lambda XY \Leftrightarrow \frac{X''}{X} + \frac{Y''}{Y} = \lambda, \tag{14}$$

enforcing that for some constants α and β,

$$X'' = \alpha X \text{ and } Y'' = \beta Y. \tag{15}$$

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

$$\lambda = (k\pi)^2 + (l\pi)^2, \quad k, l = 1, 2, 3, ... \tag{16}$$

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

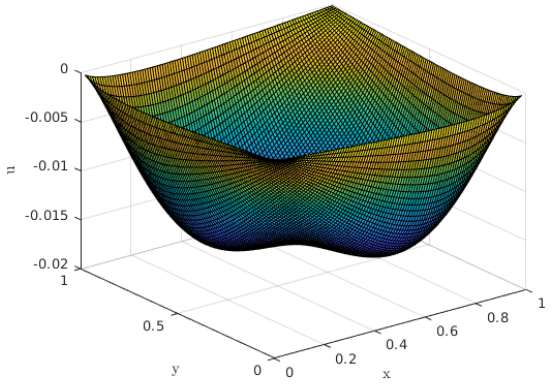
(c)

Letting

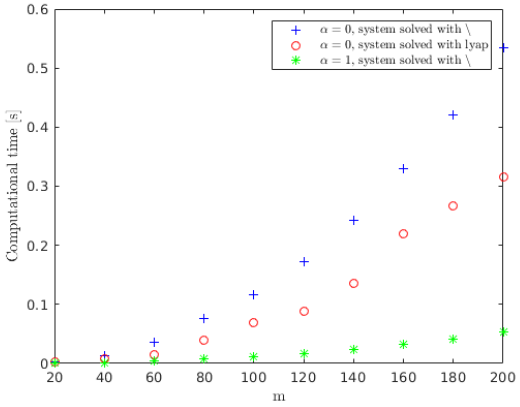
$$g(x, y) = \alpha \sqrt{(x - \frac{1}{2})^2 + (y - \frac{1}{2})^2} \tag{17}$$

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



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



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

Figure 3: Tasks 1-3

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

   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.

Fredrik: vad tror du om denna slsutsats?

(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,

$$(A - E)^{-1} = A^{-1} - A^{-1}EA^{-1} + \mathcal{O}(\|E\|^2), \tag{18}$$

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

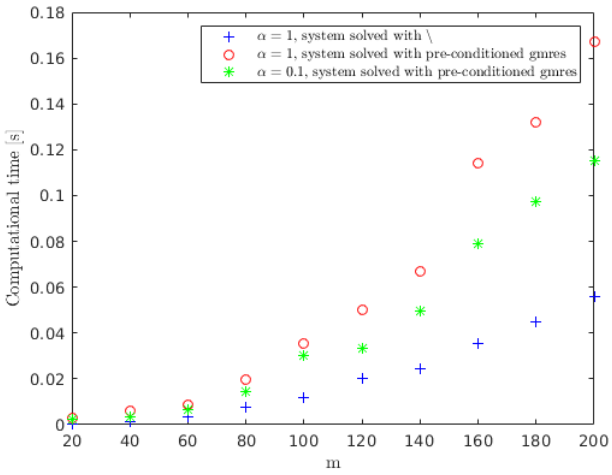


Figure 4: Timing results comparing the preconditioned gmres for different α and also comparing to results using \.

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 C is an invertible square matrix and u, v are column vectors of the same size, then $C+uv^T$ is invertible iff $1 + v^T C^{-1}u \neq 0$. The inverse is given by

$$(C + uv^T)^{-1} = C^{-1} - \frac{C^{-1}uv^T C^{-1}}{1 + v^T C^{-1}u}. \tag{19}$$

Using $u = e_{m/4}$ and $v = (1/h)e_{m/2}$, we can then compute

A naive approach representing G as a matrix to form the matrix A described in task (a), and then using \ to solve yields