# SF3580
# HW 3

Anna Broms & Fredrik Fryklund

2018/11/29

## **Task** 2

## **Task** 4

## **Task** 5

### (a)

The matrix $A$ is diagonalizable with the eigendecomposition $A = QDQ^{-1}$, where $D$ is a diagonal matrix. For such structures it holds that $\sin(A) = Q\sin(D)Q^{-1}$. Thus we can validate the result for the Schur-Parlett method, which is

$$\sin(A) = \sin\left(\begin{bmatrix} 1 & 4 & 4 \\ 3 & -1 & 3 \\ -1 & 4 & 4 \end{bmatrix}\right) \approx \begin{bmatrix} 0.846192 & 0.0655435 & -0.187806 \\ 0.33476 & 0.385017 & -0.141244 \\ -0.190921 & 0.192478 & 0.848269 \end{bmatrix}. \tag{1}$$

which in norm differs $4.28e-16$ from $Q\sin(D)Q^{-1}$.

```
using LinearAlgebra
function schur_parlett(A,f)
    T,Q,ev=schur(A)
    n = size(A,1)
    F = zeros(n,n)
    for i=1:n
        F[i,i]=f(T[i,i])
    end
    for p=1:n-1
        for i=1:n-p
            j=i+p
            s=T[i,j]*(F[j,j]-F[i,i])
            for k=i+1:j-1
                s = s + T[i,k]*F[k,j]-F[i,k]*T[k,j];
            end
            F[i,j]=s/(T[j,j]-T[i,i])
        end
    end
    F=Q*F*Q';
    return F
end
```

### (b) & (c)

It is clear from Figure 1 that the number of flops required for Schur–Parlett is not discernibly affected by $N$, at leat for $N \in 10, 50, 100, 150, 200, 250, 300$. This is not suprsining, as often the most computationally demanding part

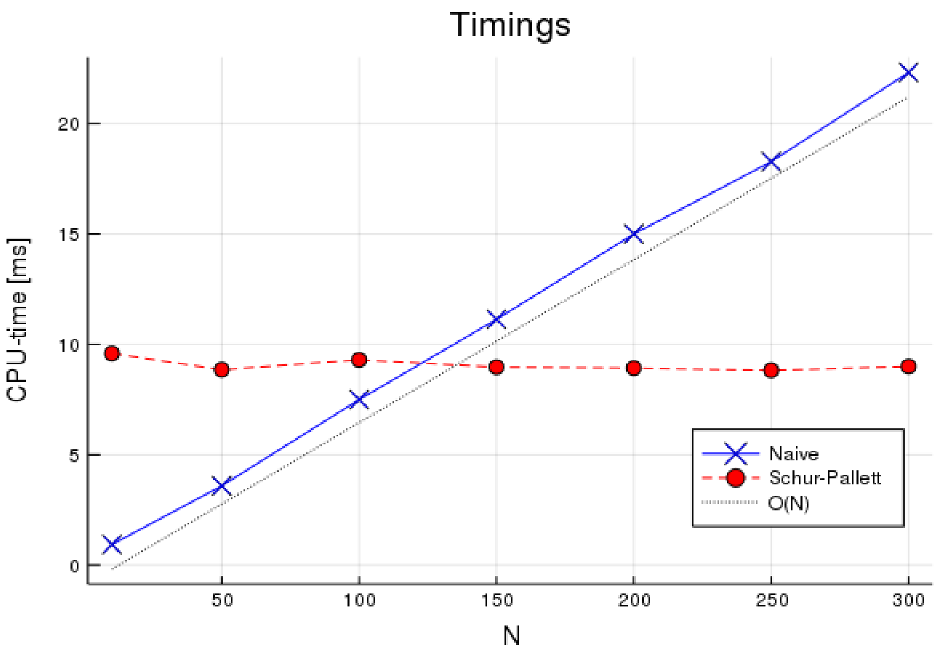


Figure 1: Task 5, (b) & (c): CPU–time in milliseconds, as a function of $N$.

of the Schur–Parlett method, in is performing the Schur decomposition, which scales like $O(n^3)$. Once obtained, the function $f$ is only applied to the diagonal elements, which are scalars.

For the naive appraoch the number of flops is proportional to $N$. A matrix multiplication is of $O(n^3)$, thus performing $N$ matrix gives $O(Nn^3)$, which we read from Figure 1. The black line corresponds to the line $0.08 + 0.07\,N$.

# Task 6

## (a)

The matrix

$$A = \begin{bmatrix} \pi & 1 \\ 0 & \pi + \varepsilon \end{bmatrix}, \tag{2}$$

with $\varepsilon > 0$, has two eigenvalues: $\lambda_1 = \pi$ and $\lambda_2 = \pi + \varepsilon$. Let $A = X\,\mathrm{diag}(J_1, J_2)\,X^{-1}$, be the Jordan canonical form, with $J_1 = \lambda_1$ and $J_2 = \lambda_2$.

The Jordan canonical form definition gives that

$$p(A) = X\,\mathrm{diag}(p(J_1), p(J_2))\,X^{-1}. \tag{3}$$

A simple consequence is

$$g(A) = X\,\mathrm{diag}(g(\lambda_1), g(\lambda_2))\,X^{-1} = X\,\mathrm{diag}(p(\lambda_1), p(\lambda_2))\,X^{-1} = p(A), \tag{4}$$

since the polynomial $p$ interpolates the function $g$ in the eigenvalues of $A$, i.e. $p(\lambda_1) = g(\lambda_1)$ and $p(\lambda_2) = g(\lambda_2)$.

Two points defines a unique polynomial of order 1, thus we may choose a $p$ in $\mathbb{P}^1$ and write $p(z) = \alpha + \beta z$. The unknown coefficents are obtianed by solving

$$\begin{cases} \alpha + \beta\lambda_1 & = g(\lambda_1) \\ \alpha + \beta\lambda_2 & = g(\lambda_2) \end{cases} \Leftrightarrow \begin{cases} \alpha & = \frac{g(\lambda_1)\lambda_2 - g(\lambda_2)\lambda_1}{\lambda_2 - \lambda_1} \\ \beta & = \frac{g(\lambda_2) - g(\lambda_1)}{\lambda_2 - \lambda_1} \end{cases} \tag{5}$$

### (b)

Given $g \coloneqq \exp$ we have from $(a)$ and $(b)$ that

$$p(A) = \alpha I + \beta A = \frac{\exp(\pi)(\pi+\varepsilon) - \exp(\pi+\varepsilon)\pi}{\varepsilon} I + \frac{\exp(\pi+\varepsilon) - \exp(\pi)}{\varepsilon} A \tag{6}$$

$$= \frac{\exp(\pi)}{\varepsilon}(\varepsilon I + (1 - \exp(\varepsilon))(\pi I - A)) \tag{7}$$

### (c)

It is known that the Jordan decomposition is unstable for non-symmetric matrices, as the eigenvalues may lie close to each other. For the given matrix $A$ this can be tuned artificially by setting $\varepsilon = |\lambda_1 - \lambda_2|$. As we see in Figure (to be added, some layout to fix). However, the function `exp` in Julia is analogous to `expm` in Matlab, (we compared the resluts for $\varepsilon = 1e-1, \dots, 1e-10$). Using Matlabs `expm`$(A)$ as reference value, the error from the exact result in (b) also increases as $\varepsilon$. This is most likely due to cancelation.

## Task 7