MAT 423: Homework 6 (10/17)

Ex 1. [SOR method]
Implement the SOR algorithm, i.e.

- input: A, ω, $\mathbf{b}$, x_0, and K the number of iterations.
- compute $(D-\omega L)^{-1}$ and $(1-\omega)D+\omega U$ and iterate:

$$\mathbf{x}_{k+1} = (D-\omega L)^{-1}\Big((1-\omega)D+\omega U\Big)\mathbf{x}_k + \omega(D-\omega L)^{-1}\mathbf{b}$$

- output: $\mathbf{x}_K$, errors $\|A\mathbf{x}_k - \mathbf{b}\|$, the spectrum of $J_\omega = (D-\omega L)^{-1}\Big((1-\omega)D+\omega U\Big)$.

Hint: see codes in next pages.

Ex 2.
Consider A the 10Ö10 tridiagonal matrix:

$$A = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}.$$

a) Compute numerically the spectrum of $J_\omega = (D-\omega L)^{-1}\Big((1-\omega)D+\omega U\Big)$ for ω varying from 0 to 2.

b) Apply the SOR method for $\omega = \frac{1}{2}$, $\omega = 1$, $\omega = \frac{3}{2}$ (choose any vectors $\mathbf{b}$ and $\mathbf{x}_0$). Plot the decay of the error versus the number of iterations.

Extra) Estimate the decay rate of the SOR methods for the three choices of ω and compare with the spectrum of J_ω.

Ex 3. [Conjugate gradient method]
Consider the matrix A defined in **Ex 2**.

a) Starting from the canonical basis $\mathbf{e}_1, \dots, \mathbf{e}_{10}$, construct a set of vectors $\mathbf{u}_1, \dots \mathbf{u}_{10}$ satisfying:

$$\langle A\mathbf{u}_i, \mathbf{u}_j \rangle = 0 \qquad \text{for any } i \neq j.$$

Hint: use the Gram-Schmidt decomposition: $\mathbf{u}_1 = \mathbf{e}_1$, *then*

$$\begin{aligned}
\mathbf{u}_2 &= \mathbf{e}_2 - \langle A\mathbf{e}_2, \mathbf{u}_1 \rangle \mathbf{u}_1 / \langle A\mathbf{u}_1, \mathbf{u}_1 \rangle, \\
\mathbf{u}_3 &= \mathbf{e}_3 - \langle A\mathbf{e}_3, \mathbf{u}_1 \rangle \mathbf{u}_1 / \langle A\mathbf{u}_1, \mathbf{u}_1 \rangle - \langle A\mathbf{e}_3, \mathbf{u}_2 \rangle \mathbf{u}_2 / \langle A\mathbf{u}_2, \mathbf{u}_2 \rangle, \\
\mathbf{u}_4 &= \mathbf{e}_4 - \dots
\end{aligned}$$

b) Take $\mathbf{x}_0 = \mathbf{b} = (1, 0, \dots, 0)^T$. Use the vectors $\{\mathbf{u}_i\}_i$ to implement the conjugate gradient method:

$$\mathbf{x}_k = \mathbf{x}_{k-1} - \langle A\mathbf{x}_{k-1} - \mathbf{b}, \mathbf{u}_k \rangle \frac{\mathbf{u}_k}{\langle A\mathbf{u}_k, \mathbf{u}_k \rangle} \qquad k = 1 \dots 10$$

Compute the error $\|A\mathbf{x}_k - \mathbf{b}\|$ for all k.

```julia
####################################
##        Hint       (Julia)       ##
####################################
using LinearAlgebra

# generate matrix A
N = 10
D = diagm(0=>2*ones(N))
U = diagm(1=>ones(N-1))
L = diagm(-1=>ones(N-1))
A = D-L-U
# compute Jω
ω = 1.5
D = diagm(0=>diag(A))
L = -(LowerTriangular(A)-D)
U = -(UpperTriangular(A)-D)
Jω = inv(D-ω*L)*((1-ω)*D + ω*U)
```

```python
####################################
##        Hint       (Python)      ##
####################################
import numpy as np

# generate the matrix A
N = 10
D = np.diag(2*np.ones(N))
U = np.diag(np.ones(N-1),1)
L = np.diag(np.ones(N-1),-1)
A = D-L-U
# compute J_omega
omega = 1.5
D = np.diag(np.diag(A))
L = -np.tril(A,-1)
U = -np.triu(A,1)
J_omega = np.linalg.inv(D-omega*L)*((1-omega)*D + omega*U)
```

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%      Hint    (Octave/Matlab)    %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% generate the matrix A
N = 10;
D = diag(2*ones(1,N));
U = diag(ones(1,N-1),1);
L = diag(ones(1,N-1),-1);
A = D-L-U;
```

```
% compute J_omega
omega = 1.5;
D = diag(diag(A));
L = -tril(A,-1);
U = -triu(A,1);
J_omega = inv(D-omega*L)*((1-omega)*D + omega*U);
```