Lecturer: Barbara Re **TAs:** Davide Torlo, Stefano Colombo

Homework 5, Deadline: 23:00, Nov 25, 2019

Numerical Methods in Informatics (L + E)

Submission info:

- for **Exercise 1**: submit two files named `powershift.m` and `SL_eigen.m` as Exercise sheet 9,
- for **Exercise 2**: submit two files named `Broyden.m` and `test_Broyden.m` as Exercise sheet 10.
- no submissions for **Exercise 3**.

Exercise 1

Points 5

The one-dimensional Sturm-Liouville eigenvalue problem models the vibration of a homogeneous string of length π that is fixed at both ends. Its discrete version, obtained by using finite difference formulas and proper boundary conditions, can be expressed by the following eigenvalue problem

$$T_n \mathbf{x} = \lambda \mathbf{x} \tag{1}$$

where T_n is the one-dimensional Poisson matrix:

$$T_n = \frac{(n+1)^2}{\pi^2} \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix} \in \mathbb{R}^{n \times n}.$$

The eigenvalues of the matrix T_n can be computed explicitly, and they are:

$$\lambda_k^{(n)} = \frac{(n+1)^2}{\pi^2} \left(2 - 2\cos\left(\frac{k\pi}{n+1} \right) \right) \qquad \text{for } k = 1, \ldots, n$$

and $0 < \lambda_1 < \lambda_2 < \cdots < \lambda_{n-1} < \lambda_n$. We want to compute the smallest eigenvalue λ_1 and corresponding eigenvector $\mathbf{x}_1$ for the case $n = 40$, with a variant of the power method.

(a) Write a function called `powershift.m` that implements the *power method with shift* (not inverse) for the computation of the largest eigenvalue and its associated eigenvector of a general matrix $A - \alpha I$, according to the following indications.

INPUT:	`A`	square matrix
	`alpha`	shift
	`tol`	tolerance ϵ to stop the method when $\lvert\lambda^{(k)} - \lambda^{(k-1)}\rvert < \epsilon \lvert\lambda^{(k)}\rvert$
	`itMax`	maximum number of iterations
	`x0`	initial guess
OUTPUT:	`lambda`	the approximation of λ_1
	`xx`	the approximation of $\mathbf{x}_1$
	`iter`	performed iterations
	`lambdas`	all computed approximations of λ_1

(b) If an upper bound τ for the largest eigenvalue λ_n of T_n is known, then we can compute the smallest eigenvalue λ_1 as the largest eigenvalue of the transformed matrix $\hat{T}_n = T - \tau I_n$ (where I is the identity matrix of size n). Write a script `SL_eigen.m` that applies the function `powershift.m` to solve the eigenvalue problem (1), with $n = 40$ and the shift $\tau = 4\frac{(n+1)^2}{\pi^2}$. Set `tol=1.0d-10` and `itMax=2500`, and use as initial guess $\mathbf{x}^{(0)} = [1, 2, \ldots, n]^{\mathrm{T}}$.

(c) In the same script `SL_eigen.m`, add the commands to plot the error on the eigenvalue $|\lambda^{(k)} - \lambda_1|$ (in semi-log scale), along with the theoretical convergence rate, which is the *square* of ratio of two largest eigenvalues in magnitude (*Hint:* pay attention to which eigenvalues play the role in this power method with shift).

(d) **Additional Point (not to be submitted):** modify the function `powershift.m` so that it returns also the history of the computed eigenvector $\mathbf{x}_1^{(k)}$. Plot the error on the eigenvalue $\|\mathbf{x}_1^{(k)} - \mathbf{x}_1\|_2$ along with the theoretical convergence rate. The exact vector $\mathbf{x}_1$ is

$$\mathbf{x}_1 = \sqrt{\frac{2}{n+1}} \left[\sin\left(\frac{\pi}{n+1}\right), \sin\left(2\frac{\pi}{n+1}\right), \ldots, \sin\left(n\frac{\pi}{n+1}\right) \right]^{\mathrm{T}} .$$

Hint: to build $\mathbf{x}_1$ in MATLAB, you can apply the function `sin` directly to a vector.

(e) **Additional Point (not to be submitted):** plot the Gerschgorin's circles to verify that the given τ is an appropriate shift.

Exercise 2

Points 5: 2(a) + 2(b) + 1(c)

The Broyden method is a variant of the Newton's method to solve non-linear systems. When multiple solutions exist, to find all of them we have to spread the initial guess around the domain. We want to find the zeros of the system of equations

$$\mathbf{f}\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{cases} x_1^2 + 4x_2^2 - 1 & = 0 \\ (x_1 - 0.5)^2 - x_2 - 1 & = 0 \end{cases}, \qquad \text{with } J_{\mathbf{f}} = \begin{bmatrix} 2x_1 & 8x_2 \\ 2(x_1 - 0.5) & -1 \end{bmatrix}. \tag{2}$$

(a) Write a function called `Broyden.m` that implements the *Broyden method* (reported below) for the computation of the zeros of a system. Given the function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$, a starting point $\mathbf{x}_0 \in \mathbb{R}^n$ and an initial guess for the Jacobian $B_0 \in \mathbb{R}^{n \times n}$, it finds the zero of the system $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, according to the following indications

INPUT:	`F`	handle function
	`x0`	initial point guess
	`B0`	initial Jacobian guess
	`tol`	tolerance ϵ to stop the method when $\|x^{(k)} - x^{(k-1)}\| + \|F(x^{(k)})\| < \epsilon$
	`itMax`	maximum number of iterations
OUTPUT:	`x`	the approximation of the zero
	`iter`	performed iterations

Broyden method:
given $\mathbf{x}^{(0)} \in \mathbb{R}^n$, $B^{(0)} \in \mathbb{R}^{n \times n}$
for $k = 1, 2, \ldots$ until stop
  solve $B^{(k)}\mathbf{p}^{(k)} = -\mathbf{f}(\mathbf{x}^{(k)})$
  set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{p}^{(k)}$
  set $\boldsymbol{\delta}\mathbf{f}^{(k)} = \mathbf{f}(\mathbf{x}^{(k+1)}) - \mathbf{f}(\mathbf{x}^{(k)})$
  compute $B^{(k+1)} = B^{(k)} + \dfrac{(\boldsymbol{\delta}\mathbf{f}^{(k)} - B^{(k)}\mathbf{p}^{(k)})\mathbf{p}^{(k)\,\mathrm{T}}}{\mathbf{p}^{(k)\,\mathrm{T}}\mathbf{p}^{(k)}}$

(b) The given system of equations (2) has two zeros, $\mathbf{x}_a^*$ and $\mathbf{x}_b^*$. Write a script `test_Broyden.m` that applies the function `Broyden.m` to find them, by using two different starting points, $\mathbf{x}_{0,a} = [-1, 0]^{\mathrm{T}}$ and $\mathbf{x}_{0,b} = [-1, -1]^{\mathrm{T}}$, respectively. Use as initial guess B_0 the exact Jacobian computed in each $\mathbf{x}_0$, i. e., $B_0 = J_{\mathbf{f}}(\mathbf{x}_0)$. Use at most 100 iterations and a tolerance $\epsilon = 10^{-10}$. Plot the two points on the same figure as red and a blue star, respectively. Save the figure as `zeros_Broyden.fig`.

(c) Help: *the attached template* `help_conv_study.m` *may guide you in solving this task.*
We want to plot on the domain $[-3, 3] \times [-2, 2]$ the convergence point of the method. Add to the same script `test_Broyden.m` the following instructions:

- save two zeros $\mathbf{x}_a^*$ and $\mathbf{x}_b^*$ computed in **(b)** in two variables;
- create a *meshgrid* of the domain with 100 points on the x–axis and 50 on the y–axis, by using the following commands

```
nx=100; ny=50;
xx=linspace(-3,3,nx);
yy=linspace(-2,2,ny);
[XX,YY] = meshgrid(xx,yy);
```

- for every point of the domain `[XX(i,j);YY(i,j)]`, use the Broyden method with initial guess $\mathbf{x}_0$ the point of the mesh and initial Jacobian guess B_0 the exact Jacobian in that point. Use only 20 maximum iterations and the tolerance 10^{-10}. Define a matrix `COL` with the same size of the meshgrid, and for each point definite it as 0 if it does not converge, 1 if it converges towards the first zero ($\|\mathbf{x} - \mathbf{x}_a^*\| \leq 10^{-8}$) or 2 if it converges towards the second zero ($\|\mathbf{x} - \mathbf{x}_b^*\| \leq 10^{-8}$).
- Finally, plot the results of this convergence study as a contour plot by using the command `contourf(XX,YY,COL)`. Add the color bar to the plot with the command `colorbar`. Now, you can see with different colors where the method will converge.
- Save the figure as `Broyden_convergence.fig`.

(d) Additional Point (not to be submitted): Repeat the last point with the identity matrix as initial guess for the Jacobian.

Additional Exercise (not to be submitted)

We consider now the QR algorithm for the computation of the spectrum applied to a real, symmetric matrix, A, with distinct eigenvalues $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_{n-1}| > |\lambda_n|$. In this specific case, the QR iterates $A^{(k)}$ converge to a diagonal matrix D whose entries are the eigenvalues of A.

(a) Implement the QR iteration for the computation of the spectrum of a square, symmetric, matrix A in a function `qr_iter` according to the following indications:

- it should check for the symmetry of the input matrix A;
- it can use the MATLAB function `qr`;
- it provides the spectrum of the matrix A in a vector;
- the stopping criteria is based on the maximum (in magnitude) of the elements below the main diagonal, i.e., of the strictly lower triangular part of the matrix $A^{(k)}$;
- it should return also the history of the errors $\{\|\Lambda^{(k)} - A^{(k)}\|_2\}_{k=1}^{\texttt{it}}$, where $\Lambda^{(k)}$ is a diagonal matrix with the approximation of the eigenvalues computed at iteration k and `it` is the last performed iteration. This quantity is a vector.

INPUT:	`A`	square matrix A
	`tol`	tolerance for the stopping criteria
	`maxIt`	maximum number of iterations
OUTPUT:	`lambdas`	vector containing the spectrum of the matrix
	`it`	performed iterations
	`Ak`	resulting matrix
	`errLambda`	the vector of errors $\{\|\Lambda^{(k)} - A^{(k)}\|_2\}_{k=1}^{\texttt{it}}$

(b) Write a script `test_qr.m` to test the function `qr_iter` on the matrix

$$A = \begin{bmatrix} 4 & 0 & -1 & 1 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 2 & 0 \\ 1 & -1 & 0 & 3 \end{bmatrix}$$

with $\texttt{tol} = \texttt{1.0e} - \texttt{8}$ and $\texttt{maxIt} = \texttt{200}$.

(c) For a symmetric positive matrix A, the following estimate of the convergence rate holds

$$\|\Lambda^{(k)} - A^{(k)}\|_2 \leq \rho \|\Lambda^{(k-1)} - A^{(k-1)}\|_2 \qquad \text{with } \rho = \max_{j=1,n-1} \frac{|\lambda_{j+1}|}{|\lambda_j|} < 1 \tag{3}$$

In the same script `test_qr.m`, add the commands to

- compute the value of ρ with the approximated spectrum of A,
- plot the error $\|\Lambda^{(k)} - A^{(k)}\|_2$ (in logarithmic scale),
- and plot, in the same figure, the expected convergence rate given by Eq. (3).