Numerical methods in scientific computing 2021

Exercise 3
Return by Tuesday 9.2.2021 23:59 to Moodle
Exercise session: Thursday 11.2.2021

Problem 1. *(pen and paper)* (6 points)

One way to determine all eigenvalues of a (small) symmetric matrix is the Jacobi method. In this method a *symmetric real* $N\times N$ matrix $\mathbf{A}$ is iteratively transformed by so called Jacobi transformations to obtain the eigenvalues to the diagonal

$$\mathbf{A}^{(i+1)} \leftarrow \mathbf{Q}_{pq}^T \mathbf{A}^{(i)} \mathbf{Q}_{pq}, \mathbf{A}^{(1)} = \mathbf{A} \qquad (1)$$

where the rotation matrix $\mathbf{Q}_{pq}$ has the form[1]

$$\mathbf{Q}_{pq} = \begin{bmatrix} 1 & & & & & & & & \\ & 1 & & & & & & & \\ & & c & & & & -s & & \\ & & & 1 & & & & & \\ & & & & 1 & & & & \\ & & & & & 1 & & & \\ & & s & & & & c & & \\ & & & & & & & 1 & \\ & & & & & & & & 1 \end{bmatrix}.$$

I.e. it is a unit matrix except that[2]:

$$(\mathbf{Q}_{pq})_{pp} = (\mathbf{Q}_{pq})_{qq} = c$$

and

$$(\mathbf{Q}_{pq})_{pq} = -s\ ,\ (\mathbf{Q}_{pq})_{qp} = s\ .$$

The algorithm iteratively applies the transformation (1). Expressions for the nonzero elements are the following[3]:

$$c = \sqrt{\frac{1+C}{2}}\ ,$$
$$s = \operatorname{sign}(a_{pq}) \sqrt{\frac{1-C}{2}}\ ,$$

1 Note that here the subscript pair *pq* does not indicate a matrix element but specifies to which rows and columns in matrix **A** are transformed.
2 Now the outer subscripts denote the matrix element.
3 a_{pq} denotes the element pq of matrix **A**.

$$C = \frac{a_{pp} - a_{qq}}{\sqrt{(a_{pp} - a_{qq})^2 + 4a_{pq}^2}} \, .$$

This is so called Givens rotation where the off-diagonal elements (a_{pq} , a_{qp}) of **A** are zeroed without touching the other elements. While the successive iterations spoil the previous zeroing-out of the off-diagonal elements one can show that the diagonal elements of the iterated matrix approach the eigenvalues of the original matrix **A** . At the same time the off-diagonal elements die out.

The iteration proceeds as follows:

set $\mathbf{A}^{(1)} \leftarrow \mathbf{A}$
for $i = 1 : i_{\max}$ {
  set $\mathbf{B} \leftarrow \mathbf{A}^{(i)}$
  for $p = 1 : N - 1$
    {
    for $q = p + 1 : N$
      {
      compute *c*, *s*
      construct $\mathbf{Q}_{pq}$
      set $\mathbf{B} \leftarrow \mathbf{Q}_{pq}^T \mathbf{B} \mathbf{Q}_{pq}$
      }
    }
  set $\mathbf{A}^{(i+1)} \leftarrow \mathbf{B}$
}

Show that matrix $\mathbf{Q}_{pq}$ is orthogonal.

Problem 2. *(Computer)* (12 points)

A) Write a function `jacobi(Q,N)` that computes eigenvalues of an $N \times N$ real matrix Q implementing the Jacobi method. Put your function in a file named “`jacobi`”. You cannot use any library implementations of the Jacobi method.
B) Check that it works for symmetric matrices by comparing with results obtained from Matlab, Numpy or LAPACK library functions. Are there any differences in the results obtained by your Jacobi function and the library you used? Explain why.

Problem 3. (*Computer*) (6 points)

In this problem you will experiment with the error propagation in the eigenvalue method of the previous problem.

A) Write a function `err_propag(N,dq)` which calculates the error propagation of the eigenvalue problem. Put your function in the same file “`jacobi`” after `jacobi(Q,N)`. Your function must:
  - Define a random symmetric $N \times N$ matrix (you can use any random

number generator library subprogram) and use your Jacobi function[1] to calculate its eigenvalues λ_i .

- Randomly select an element and perturb it by a relative error dq . Then recalculate the eigenvalues $\lambda_i^{(p)}$ of the perturbed matrix.
- Calculate the error $\delta\lambda_i = \lambda_i^{(p)} - \lambda_i$ caused by the perturbation.
- Return the relative error propagation factor $f = \frac{\|\delta\lambda_i\|}{\|\lambda_i\| dq}$ (relative error in the output divided by relative error of the input). Use Euclidean norms.

B) Call your function many times (of the order of 100) for various system sizes (of the order of 10×10) and various perturbation levels dq (of the order of 1%) . Collect the statistics for *f* and determine its mean value and standard deviation.

1 If you have not completed successfully Problem 2, you can use a library implementation to calculate the eigenvalues.