Prof. Dr. U. Rüde
Benjamin Mann

Algorithms of Numerical Linear Algebra
Assignment 5

Exercise 1 (*A Singular System*) **3P.**

Let $A \in \mathbb{C}^{m \times m}$ be Hermitian positive semidefinite with $\mathrm{rank}(A) = m - 1$ and let $b \in \mathrm{range}(A)$. Note that the system $Ax = b$ is not uniquely solvable. Our aim is to find the solution $x \in \mathbb{C}^m$ which is orthogonal to $\ker(A)$. To that end, we define a nonsingular auxiliary system $(A + uu^*)y = b$.

(a) Define $u \in \mathbb{C}^m$ and prove that the resulting system is nonsingular.

(b) How is x related to y (Proof)? Also prove the orthogonality $x \perp \ker(A)$.

Exercise 2 (*Gerschgorin's Theorem*) **7P.**

Gerschgorin's theorem: *Let $A \in \mathbb{C}^{m \times m}$. Every eigenvalue of A lies in at least one of the m circular disks in the complex plane with centers a_{ii} and radii $\sum_{j \neq i} |a_{ij}|$. Furthermore, let $\mathcal{U}$ denote the union of n of these disks. If $\mathcal{U}$ is disjoint from the union of the remaining $m - n$ disks, then, exactly n eigenvalues of A are located in $\mathcal{U}$.*

(a) Prove the first part of Gerschgorin's theorem. (Hint: Let λ be any eigenvalue of A, and x a corresponding eigenvector with largest entry 1.)

(b) Prove the second part. (Hint: Decompose A into a diagonal and a remainder part and use the fact that the eigenvalues of a matrix are continuous functions of its entries.)

(c) Use Gerschgorin's theorem and your knowledge about eigenvalues (Exercise 3) to estimate the condition number $\kappa = \|A\|_2 \|A^{-1}\|_2$ for

$$A = \frac{1}{2} \begin{pmatrix} -7 & -1 & 0 \\ -1 & -3 & 1 \\ 0 & 1 & -2 \end{pmatrix}$$

Note: Give both an upper and a lower bound for κ!

Exercise 3 (*QR Algorithm*) **10P.**

Make sure to follow the requirements for programming tasks stated on the information sheet!

Step by step, we will now put together an Python program that finds all the eigenvalues of a real symmetric matrix.

(a) Write a function `T = tridiag(A)` that reduces a real symmetric $m \times m$ matrix to tridiagonal form by orthogonal similarity transformations. Your program should use only elementary

Python/numpy operations. The output matrix `T` should be symmetric and tridiagonal up to rounding errors. If you like, add a line that forces `T` at the end to be exactly symmetric and tridiagonal. Test your function for example with `A = hilbert(4)`.

(b) Write a function `(Tnew, [t]) = QR_alg(T)` that runs the unshifted QR algorithm on a real tridiagonal symmetric matrix `T`. For the QR factorization use `numpy.linalg.qr`. Again, you may wish to enforce symmetry and tridiagonality at each step. Your program should iterate until $|t_{m,m-1}| < 10^{-12}$ (hardly an industrial strength convergence criterion!). The return values are the current tridiagonal matrix `Tnew` and a list `t`, s.th. `t[i]`$= |t_{m,m-1}|$ after iteration `i` (i.e. `t[0]` corresponds to the initial matrix `T`). Again, apply your program to `A = hilbert(4)` (of course, you have to apply `tridiag` first).

(c) Write a function μ = `wilkinson_shift(A)` which computes the factor μ according to equation (29.8) from [1].

(d) Write a function `(Tnew, [t]) = QR_alg_shifted(T)` that runs the shifted QR algorithm applying the *Wilkinson shift* at each step (use the function `wilkinson_shift(A)` to compute μ). The return values are defined as in (c).

(e) Write a driver program `([Λ], [all_t]) = QR_alg_driver(A, shift)` which

 • (i) reduces `A` to tridiagonal form by calling `tridiag`

 • (ii) computes an eigenvalue of `A` by calling `QR_alg_shifted` or `QR_alg`, depending on whether `shift==True` or `shift==False`.

 • (iii) computes the next eigenvalue by applying the QR algorithm again on a smaller sub-matrix and so on until all of the eigenvalues of `A` are determined.

The return values are a list of all the eigenvalues of A and the concatenation of the lists `t` obtained from calling the QR algorithm.

(f) Run your program for the matrices

 • `hilbert(4)`

 • `np.diag([1,2,3,4]) + np.ones((4, 4))`

 • `np.diag([5,6,7,8]) + np.ones((4, 4))`

Generate a **semilogy** plot `[all_t]` over `[iterations]` for each of the 3 matrices and both `shift==True` and `shift==False`. Discuss the convergence behavior observed from the different matrices and algorithms (rates of convergence as well as constant factors). Do your observations fit the statements in the textbook?

References

[1] L.N. Trefethen and D. Bau. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997.