Fall 2017: Numerical Methods I
Assignment 7 (due Dec 14, 2017)

This assignment again contains sets of basic questions to check your understanding of the concepts from class. These questions require either a true/false, or another short answer. You should be able to do them quickly. Similar questions will also be part of the final. For each block of 10 questions, you will get 0.5 points if you have at least 6 correct answers, 1 points for 7, 1.5 for 8, 2 for 9, and 2.5 points for 10 correct answers.

1. **[Basic understanding questions, 4×2.5 points]**

Let $F : D \to \mathbb{R}^n$ with an open set $D \subset \mathbb{R}^n$, and we assume that F is continuously differentiable. We are interested in finding $\boldsymbol{x}^*$ such that $F(\boldsymbol{x}^*) = 0$.		
1	Since F is continuously differentiable, there is at least one $\boldsymbol{x}$ with $F(\boldsymbol{x}) = 0$.	
2	Newton's method converges from any initialization $\boldsymbol{x}^0$.	
3	Damping of the update step can be used for faster local convergence close to the solution.	
4	For a regular matrix $A \in \mathbb{R}^{n\times n}$, solving $AF(\boldsymbol{x}) = 0$ is equivalent to solving $F(\boldsymbol{x}) = 0$.	
5	If Newton's method converges quadratically to a solution $\boldsymbol{x}^*$, this implies that close to the solution $\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^*\| \leq c\|\boldsymbol{x}^k - \boldsymbol{x}^*\|$ with $c < 1$.	
6	Assume that F is twice differentiable and that the Jacobian $F'(\boldsymbol{x}^*)$ is invertible. Specify the highest local convergence order you can guarantee.	
7	Let $F : \mathbb{R} \to \mathbb{R}$ defined by $F(x) = \exp(x^2) - 2$. For this problem, Newton's method to solve $F(x) = 0$ converges from any initialization.	
8	Newton's method can also be considered as a fixed point method.	
9	Let $F(x) := x^4 - 2$. Then the Newton iteration to find a root of F is $x_{k+1} := \frac{3}{4}x_k - \frac{1}{x_k^3}$.	
10	Give an example for a function F, for which Newton's method to compute roots of F only converges linearly.	

Some of the below questions refer to a function $f : \mathbb{R}^2 \to \mathbb{R}$ defined by $f(x, y) = y \exp(4x^2)$, and we use the 1-norm for $\mathbb{R}^2$.		
1	There are gaps between two consecutive numbers stored in a computer.	
2	All these gaps have the same size, called the machine epsilon.	
3	The decimal number 10.25 can be represented exactly in single precission.	
4	The decimal number 10.25 can be represented exactly in double precission.	
5	In one kilobyte of memory, one can store 128 numbers in double precision or 256 numbers in single precision.	
6	The relative condition number of f is $\kappa_{rel} = 1 + 8x^2$.	
7	The relative condition number of f is $\kappa_{rel} = \max\{1, 8x^2\}$.	
8	f is poorly conditioned for $\lvert y \rvert \to \infty$.	
9	Addition of two numbers is poorly conditioned if the numbers are almost the same.	
10	Say, the condition number of a matrix A is 10^{12} and the machine epsilon is 10^{-16}. Both the matrix and a right hand side vector $\boldsymbol{b}$ have order 1 entries. How many accurate digits will the solution of $A\boldsymbol{x} = \boldsymbol{b}$ at least have?	

Let $A \in \mathbb{R}^{n \times n}$.		
1	The LU-composition $A = LU$ always exists if A is invertible.	
2	The QR-composition $A = QR$ always exists if A is invertible.	
3	The Choleski factorization $A = LL^T$ always exists if A is symmetric and positive definite.	
4	Let $\kappa(A)$ be the condition number of A. Considering a perturbation only in $\boldsymbol{b}$, then the relative error in $\boldsymbol{x}$ is at the most by $\kappa(A)$ larger than the relative error in $\boldsymbol{b}$.	
5	Multiplication of a dense matrix with a vector requires $O(n)$ flops.	
6	Solving $U\boldsymbol{x} = \boldsymbol{b}$ with an upper triangular matrix U requires $O(n^2)$ flops.	
7	Compute the 1-norm (i.e., the matrix norm induced by the $\|\cdot\|_1$ vector norm) of the matrix $A = \begin{bmatrix} -2 & 4 \\ 0 & 3 \end{bmatrix}$.	
8	Let $A = QR$ be a QR factorization. Then $\det(A) = \det(R)$.	
9	Let Q be an orthogonal matrix such that $Q \begin{bmatrix} 1 \\ -3 \end{bmatrix} = \begin{bmatrix} c \\ 0 \end{bmatrix}$. What is $\|c\|$?	
10	If the QR-factorization $A = QR$ of A is computed using Givens rotations, then $Q^{-1} = Q$.	

Let $A \in \mathbb{R}^{m \times n}$ with $n \leq m$. Let the rank of A be n and $\boldsymbol{b} \in \mathbb{R}^m$. Let $Q \in \mathbb{R}^{m \times m}$ orthogonal and $R \in \mathbb{R}^{m \times n}$ an upper triangular matrix with $A = QR$, and let $\| \cdot \|$ be the Euclidean norm.		
1	The matrix $A^T A$ is symmetric and positive definite.	
2	Denote $\boldsymbol{x}^*$ the minimizer of $\min_{\boldsymbol{x} \in \mathbb{R}^n} \|A\boldsymbol{x} - \boldsymbol{b}\|$. Then $\boldsymbol{x}^*$ satisfies $A^T A \boldsymbol{x}^* = \boldsymbol{b}$.	
3	Denote $\boldsymbol{x}^*$ as above, and let Θ be the angle between $A\boldsymbol{x}^*$ and $\boldsymbol{b}$. Then, the larger Θ, the worse the conditioning of the least squares problem.	
4	For all $\boldsymbol{x} \in \mathbb{R}^n$ holds $\|A\boldsymbol{x} - \boldsymbol{b}\| = \|R\boldsymbol{x} - Q\boldsymbol{b}\|$.	
5	The method of Givens rotations to compute the QR-factorization does not require pivoting.	
6	We know that R has the form $[R_1, 0]^T$, with $R_1 \in \mathbb{R}^{n \times n}$. It holds that $\det(A) = \det(R_1)$.	
7	For sufficiently smooth functions, the Gauss-Newton method for the solution of a nonlinear least squares problem converges locally at a quadratic rate.	
8	The Gauss-Newton method for the solution of a nonlinear least squares problem solves a linear least squares problem in each iteration.	
9	Damping the update step in the Gauss-Newton method is a method to achieve convergence for problems that might otherwise not converge.	
10	The Gauss-Newton method is identical to the Newton method for solving the first-order necessary condition of the nonlinear least squares minimization problem.	

2. **[Cubic spline construction—brute force version, 3pt]** Let us construct a cubic spline[1] for the nodes $t_0 = 0$, $t_1 = 1$, $t_2 = 2$, $t_3 = 3$. A cubic spline follows a cubic polynomial in each interval $I^{(j)} := [t_j, t_{j+1}]$, $j = 0, 1, 2$, and is twice continuously differentiable everywhere in $[t_0, t_3]$. To find the cubic spline, let us use a monomial basis in each of the intervals $I^{(j)}$:

$$s^{(j)}(t) = a_0^{(j)} + a_1^{(j)} t + a_2^{(j)} t^2 + a_3^{(j)} t^3, \qquad \text{for } j = 0, 1, 2.$$

Express the conditions the spline satisfies at the nodes t_i in terms of conditions for $s^{(j)}$ and its derivatives, and derive the resulting 10 linear conditions for the 12 coefficients $a_i^{(j)}$. To make this under-determined system uniquely solvable, either add zero conditions for the first or the second derivatives at t_0 and t_3. Let us now interpolate the function $f(t) = \cos \pi t \exp(t/2)$. Compute the spline coefficients by solving the resulting linear system[2] for both choices of boundary conditions at the first and last node. Plot your results and compare with the build-in cubic spline interpolation (in MATLAB, you can use the `interp1` function, which has a `'spline'` option—see the description of the function.) What conditions at the first and last node does the build-in function use?

3. **[Extrapolation, 3pt]** Extrapolation is interpolation where the point of interest, at which we evaluate the interpolation polynomial, is *outside* the interpolation interval.[3] Extrapolation can, for instance, be used to approximate limits. Let us compute the sum

$$s := \sum_{k=1}^{\infty} \frac{1}{k^{3/2}},$$

with has the exact value $\zeta(3/2) = 2.61237534868548834\ldots$, through extrapolation. For that purpose, consider the partial sums

$$s_i := \sum_{k=1}^{n_i} \frac{1}{k^{3/2}} \qquad \text{with } n_i = 10 \cdot 2^i, \text{ for } i = 1, \ldots, 5.$$

Denoting by $h_i := 1/n_i$, extrapolation then amounts to fitting a polynomial to the values (h_i, s_i), for $i = 1, \ldots, 5$, and evaluating that polynomial at $h = 0$ (which corresponds to $n = \infty$). The result $\tilde{s}$ is the extrapolated value for the approximation of s.

Use build-in functions for the interpolation/extrapolation (i.e., in MATLAB, the function `polyfit` and `polyval`). List the values s_i and the extrapolated value $\tilde{s}$ and highlight the number of exact digits. Note that you will have to output at least 12 digits (use `format long` in MATLAB).

---

[1] Note that there are more efficient and stable ways to construct cubic splines, namely de Boor's algorithm and its variants, which we haven't had time to cover in class. The purpose of this problem is to use a straightforward algorithm that allows us basic experiments with spline interpolation.

[2] Note that in this approach, one has to solve a linear system of size $4(n+1)$ to find a spline with $n+1$ nodes. De Boor's algorithm avoids that global solve by using a better basis that again satisfies a recursion.

[3] See Section 9.4.2 in Deuflhard/Hohmann or https://en.wikipedia.org/wiki/Extrapolation.

4. [**Trapezoidal and Simpson sum, 3+2+2pt]** Write functions `trapez(f,x)` and `simpson(f,x)` to approximate integrals

$$I(f) = \int_a^b f(t)\,dt$$

using the trapezoidal and Simpson's rules on each sub-interval $I_i := [x_i, x_{i+1}]$, $i = 0, 1, \ldots, N-1$, where we assume that these sub-intervals cover $[a, b]$ and have no overlap except the sub-interval boundary points x_i. The input vector to the functions should be $\boldsymbol{x} = (x_0, \ldots, x_N)$, i.e., the vector of sub-interval boundaries, and f is either a function handle[4] or the vector $\boldsymbol{f} = (f(x_0), \ldots, f(x_N))$. Note that `simpson()` also requires the function values at the mid points of each interval, i.e., if $\boldsymbol{f}$ is a vector, it must include the values $f((x_{i+1} + x_i)/2)$ and must thus be of size $2N + 1$.

(a) Hand in code listings of these functions, and use them to approximate the integral

$$\int_0^5 \sqrt{x}\,dx.$$

Compare the numerical errors for uniformly spaced points $x_i = 5ih$, $i = 0, \ldots, N$, $h = 1/N$ for both quadrature rules. Try different N and plot the quadrature errors versus N in a double-logarithmic plot.

(b) Estimate the error behavior by fitting cN^κ, with $c, \kappa \in \mathbb{R}$ to the quadrature errors. To avoid having to solve a nonlinear least squares problem for c and κ, apply the logarithm to cN^κ and solve a *linear* least squares problem for $d := \log(c)$ and κ.

(c) Use extrapolation as disussed in the previous problem to improve the results you find with the trapezoidal rule for the above integral. That is, compute the numerical approximations for the integral for different values of h, and then use extrapolation to estimate the corresponding value for $h = 0$. Compare the accuracy with the exact solution.

5. **[Hermite quadrature, 3+3pt]** Here, we study a quadrature rule that involves derivatives.

(a) Let us find basis polynomials $H_0(t), H_1(t), H_2(t) \in \boldsymbol{P}_2$ for Hermite interpolation of $f(0)$, $f'(0)$ and $f(1)$, i.e., find the quadratic polynomials that satisfy

$$\begin{aligned} H_0(0) &= 1, \quad H_0'(0) = 0, \quad H_0(1) = 0, \\ H_1(0) &= 0, \quad H_1'(0) = 1, \quad H_1(1) = 0, \\ H_2(0) &= 0, \quad H_2'(0) = 0, \quad H_2(1) = 1. \end{aligned}$$

Derive these polynomials using the Newton basis and the divided differences scheme.

(b) For numerical integration on the interval $[0, 1]$, find weights $\alpha_1, \alpha_2, \alpha_3$ such that the quadrature formula

$$\hat{I}(f) := \alpha_1 f(0) + \alpha_2 f'(0) + \alpha_3 f(1)$$

integrates polynomials $p \in \boldsymbol{P}_2$ exactly. *Hint:* Use that the Hermite polynomials H_0, H_1, H_2 are a basis of $\boldsymbol{P}_2$, i.e., it suffices if $\hat{I}$ is exact for these H_i.

[4] See http://www.mathworks.com/help/matlab/matlab_prog/creating-a-function-handle.html if you are not familiar with that concept.

6. [**Hierarchical Lobatto quadrature, 3+2pt]** Gauss[5] quadrature chooses the quadrature point location and the quadrature weights such that a high order approximation is achieved with a minimal number of points. The Gauss quadrature points for an interval $[a, b]$ do not include the beginning and end points a and b. It is often convenient if these points are included. When not all points are allowed to vary but the location of some points is fixed, the corresponding quadrature rules are called Gauss-Lobatto rules. Let us compute such Gauss-Lobatto points and corresponding quadrature weights for the interval $[-1, 1]$. Due to the symmetry, the quadrature formula for 4 nodes is:

$$\hat{I}_1(f) = \omega_1(f(-1) + f(1)) + \omega_2(f(-t_2) + f(t_2)),$$

with to be determined weights ω_1, ω_2 and the interior node location t_0. Because of symmetry, the monomials with odd degree are integrated exactly by the above formula. Since there are 3 parameters, we hope that we can choose them to integrate the even monomials $1, x^2, x^4$ exactly.

   (a) Give the set of nonlinear equations needed to be satisfied by ω_i, ω_2 and t_2 to integrate the first three even monomials exactly, and solve it analytically.[6]

   (b) Often, one is interested in an adaptive quadrature rule that allows to add quadrature points while re-using the quadrature points from a less accurate rule, where the function has already been evaluated.[7] Thus, let us try to extend the above quadrature rule by adding internal nodes, while re-using the point t_2. We add three points: Due to symmetry, one must lie at the center ($t = 0$), and the other two added points must be symmetric around the center, such that the new quadrature rule becomes:

$$\hat{I}_2(f) = \omega_3(f(-1) + f(1)) + \omega_4(f(-t_2) + f(t_2)) + \omega_5(f(-t_3) + f(t_3)) + \omega_6 f(0).$$

   Note that t_2 is fixed but the corresponding weight ω_4 can differ compared to the quadrature formula $\hat{I}_1$. Thus, the free variables in $\hat{I}_2$ are $\omega_3, \omega_4, \omega_5, \omega_6$ and t_3, and we can hope to integrate the monomials $1, x^2, x^4, x^6, x^8$ exactly. Specify the corresponding nonlinear system.

   (c) **[1pt extra credit]** Solve this nonlinear system for the five parameters using Newton's method.

7. **[Iterative solution of linear systems, 1+1+1+2pt]** Let us study iterative solvers for linear systems $A\boldsymbol{x} = \boldsymbol{b}$, where $A \in \mathbb{R}^{n\times n}$ and $\boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^n$. For every n, A and $\boldsymbol{b}$ have the

[5] In this example, we use "Gauss quadrature" short for Gauss-Legendre quadrature, since we use the integral weight function $\omega \equiv 1$, and the corresponding orthogonal polynomials are the Legendre polynomials.

[6] A Gauss rule with 4 points and 4 weights allows exact integration of polynomials up to order 7. This Gauss-Lobatto (or Gauss-Legendre-Lobatto) rule, which fixes two node locations, only integrates polynomials up to order 5 exactly. This, namely that Gauss quadrature is two orders more accurate than Lobatto-Gauss quadrature, remains true for higher-order quadrature.

[7] Function evaluation is usually the most expensive step in numerical quadrature, i.e., the target is to minimize the number of required evaluations of f while obtaining a good approximation to the integral.

same structure, for instance for $n = 7$:

$$A = \begin{bmatrix} 0 & 0 & 2.01 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2.01 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2.01 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2.01 & -1 \\ -1 & 0 & 0 & 0 & 0 & -1 & 2.01 \\ 2.01 & -1 & 0 & 0 & 0 & 0 & -1 \\ -1 & 2.01 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

(a) Why is it not possible to apply the Jacobi or the Gauss-Seidel iteration for solving this linear system directly? Which rows/columns must be exchanged such that the resulting linear system can be solved with these methods? In the following, we always consider the system with properly exchanged rows/columns.

(b) Argue why the Jacobi and the Gauss Seidel method converge.

(c) Specify the iteration for the Jacobi and the Gauss-Seidel iteration component-wise.

(d) Implement both methods using component-wise expressions—you should never have to assemble a matrix. Apply your implementation for the solution of the problem for $n = 10^2, 10^3, 10^4$, and report the number of iterations for each n for both methods. Use $\boldsymbol{x}_0 = 0$ as initialization, and terminate the iteration when the norm of the residual $\boldsymbol{r}_k := \boldsymbol{b} - A\boldsymbol{x}_k$ has decreased by a factor of 10^8 compared to the initial residual $\boldsymbol{r}_0$, i.e., when

$$\frac{\|\boldsymbol{r}_k\|}{\|\boldsymbol{r}_0\|} < 10^{-8}.$$