Spring 2017: Numerical Analysis
Assignment 3 (due Mar. 9, 2017)

1. **[1+2+3pt]** Computers use finite precision to represent real numbers, which leads to rounding. You can see the size of the rounding error for real numbers around 1 using the MATLAB command `eps(1)`. This number, also called the *machine epsilon*, is $\epsilon = 2.22 \times 10^{-16}\dots$ for the standard (double precision) representation of numbers in a computer. Try the following experiments in MATLAB (or Python/Octave).[1]

   (a) Report the analytical/exact result, and the result you get when using your computer for:
   $$a = (1-1) + 10^{-16}, \quad b = 1 - (1 + 10^{-16}).$$
   What do you think is happening?

   (b) The n-th Hilbert matrix $H_n \in \mathbb{R}^{n\times n}$ has the entries $h_{ij} = (i+j-1)^{-1}$ for $i,j = 1,\dots,n$.[2] It is known that solving systems with the Hilbert matrix increases rounding errors since the matrix is poorly conditioned. Let $\boldsymbol{e}_n$ be the column vector of length n that contains all 1's. Report the exact and the numerically computed values for
   $$\|H_n(H_n^{-1}\boldsymbol{e}_n) - \boldsymbol{e}_n\|, \text{ for } n = 5, 10, 20.$$
   Here, $\|\cdot\|$ is the usual Euclidean norm. Also report the condition numbers of H_n (with respect to either norm) for $n = 5, 10, 20$.

   (c) As you know, for differentiable functions f holds
   $$f'(x_0) = \lim_{h\to 0} \frac{f(x_0+h) - f(x_0)}{h}.$$
   Thus, to numerically approximate a derivative of a function for which the derivative is hard to derive analytically, one can use, for a small h, the approximation
   $$f'(x_0) \approx f'_{\text{num}}(x_0) := \frac{f(x_0+h) - f(x_0)}{h}.$$
   In this approximation, one subtracts very similar numbers in the numerator of the fraction from each other, and then multiplies with a large number (namely h^{-1}), which can lead to errors that are much larger than the machine epsilon. Compute approximations of the derivative of the function
   $$f(x) = \frac{\exp(x)}{\cos(x)^3 + \sin(x)^3}$$
   at $x_0 = \pi/4$. In order to do so, use progressively smaller perturbations $h = 10^{-k}$ for $k = 1, \dots 16$. Present the errors in the resulting approximation in a log-log plot,

[1]You can use the command `format long` to get 15 digits output from your computer. If you need more digits of a number a, you can use `fprintf('%2.20f\n',a)` to see 20 digits.

[2]You can get H_n by using the MATLAB command `hilb(n)`.

i.e., use a logarithmic scale to plot the values of h on the x-axis, and a logarithmic scale to plot the errors between the finite difference approximation $f'_{\text{num}}(x_0)$ and the exact value, which is $f'(x_0) = 3.101766393836051$, on the y-axis.[3] What do you observe as h becomes smaller, and for which h do you get the best approximation to the derivative?

2. **[2+1+2pt+2pt (extra credit)]** Let us explore matrix norms and condition numbers.

   (a) For the following matrix given by

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

   calculate $\|A\|_1$, $\|A\|_2$, $\|A\|_\infty$ as well as the condition numbers for each norm by hand. Is A well or ill-conditioned?

   (b) Recall the formulas from Theorems 2.7 and 2.8. If you assume that taking the absolute value and determining the maximum does not contribute to the overall computational cost, how many *flops* (floating point operations) are needed to calculate $\|A\|_1$ and $\|A\|_\infty$ for $A \in \mathbb{R}^{n\times n}$? By what factor will the calculation time increase when you double the size of matrix size?

   (c) Now implement a simple code that calculates $\|A\|_1$ and $\|A\|_\infty$ for a matrix of any size $n \geq 1$. Try to do this without using loops[4]! Using system sizes of $n_1 = 100$, $n_{k+1} = 2 * n_k, k = 1..7$, determine how long your code takes [5] to calculate $\|A\|_1$ and $\|A\|_\infty$ for a matrix $A \in \mathbb{R}^{n_i \times n_i}$ with random entries and report the results. Can you confirm the estimate from (b)?

   (d) **(extra credit)** MATLAB has the build-in function `norm` to calculate matrix norms.[6] Calculate for the system sizes in (c) $\|A\|_1$ and $\|A\|_\infty$ using both your implementation and MATLAB's `norm` function, determine for each n_i how long each code takes and plot the results in one graph. On average, by what factor is MATLAB's implementation faster than yours?

   Please also hand in your code.

3. **[4pt]** Show that, for any $\boldsymbol{v} \in \mathbb{R}^n$, we have

   $$\|\boldsymbol{v}\|_\infty \leq \|\boldsymbol{v}\|_2 \quad \text{and} \quad \|\boldsymbol{v}\|_2^2 \leq \|\boldsymbol{v}\|_1 \|\boldsymbol{v}\|_\infty.$$

   In each case, give an example of a nonzero $\boldsymbol{v}$ for which equality is obtained.

[3] MATLAB offers a `loglog` function to do that.

[4] The commands needed in MATLAB are `abs` and `sum`. Most commands can not only applied to numbers, but also to vectors, where they apply to each component.

[5] In MATLAB use the *stop watch* commands `tic` and `toc`.

[6] Use `help norm` to find out how to obtain the matrix norm that is induced by either the 1,2 or ∞-vector norm.

4. **[4pt]** We can define the $\|\cdot\|_p$ matrix norms ($p \in \{1, 2, \infty\}$) for an $m \times n$ matrix A by

$$\|A\|_p = \sup_{\boldsymbol{v}\in\mathbb{R}^n\setminus\{\mathbf{0}\}} \frac{\|A\boldsymbol{v}\|_p}{\|\boldsymbol{v}\|_p}$$

where the norm in the numerator is defined on $\mathbb{R}^m$ and the norm in the denominator is defined on $\mathbb{R}^n$.
Using the problem above, show that

$$\|A\|_\infty \le \sqrt{n}\|A\|_2 \quad \text{and} \quad \|A\|_2 \le \sqrt{m}\|A\|_\infty$$

In each case, give an example of a nonzero matrix A for which equality is obtained.

5. **[3pt]** Let $A \in \mathbb{R}^{n\times n}$, let λ be an eigenvalue of A^TA and $\boldsymbol{x} \in \mathbb{R}^n\setminus\{\mathbf{0}\}$ be the corresponding eigenvector.

   (a) Show that $\|A\boldsymbol{x}\|_2^2 = \lambda\|\boldsymbol{x}\|_2^2$ and hence that $\lambda \ge 0$.

   (b) Let $\|\cdot\|$ be a norm on $\mathbb{R}^n$ with associated subordinate matrix norm $\|\cdot\|$ on $\mathbb{R}^{n\times n}$. Show that $\|A\|_2 \le \|A^TA\|^{1/2}$ (Hint: first show that $\lambda \le \|A^TA\|$)

   (c) Using part (b) with matrix norm $\|\cdot\|_1$, show that $\kappa_2(A) \le (\kappa_1(A)\kappa_\infty(A))^{1/2}$

6. **[2pt]** Let $A \in \mathbb{R}^{n\times n}$ be invertible. Let $\boldsymbol{b} \in \mathbb{R}^n\setminus\{\mathbf{0}\}$, and $A\boldsymbol{x} = \boldsymbol{b}$, $A\boldsymbol{x}' = \boldsymbol{b}'$ and denote the perturbations by $\Delta\boldsymbol{b} = \boldsymbol{b}' - \boldsymbol{b}$ and $\Delta\boldsymbol{x} = \boldsymbol{x}' - \boldsymbol{x}$. Show that the inequality obtained in Theorem 2.11 is *sharp*. That is, find vectors $\boldsymbol{b}, \Delta\boldsymbol{b}$ for which

$$\frac{\|\Delta\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} = \kappa_2(A)\frac{\|\Delta\boldsymbol{b}\|_2}{\|\boldsymbol{b}\|_2} .$$

(Hint: consider the eigenvectors of A^TA.)

7. **[3pt]** Using the `qr` function in MATLAB (or however else you like), find the QR factorization of

$$A = \begin{bmatrix} 9 & -6 \\ 12 & -8 \\ 0 & 20 \end{bmatrix} .$$

Write down both formulations we discussed in class, i.e., $A = \hat{Q}\hat{R}$ with $\hat{Q} \in \mathbb{R}^{m\times n}$, $\hat{R} \in \mathbb{R}^{n\times n}$ as well as $A = QR$ with $Q \in \mathbb{R}^{m\times m}$, $\hat{R} \in \mathbb{R}^{m\times n}$. Use it to find the least squares solution to the system of linear equations

$$\begin{aligned} 9x - 6y &= 300 \\ 12x - 8y &= 600 \\ 20y &= 900 . \end{aligned}$$

Plot the three lines above and indicate the location of the least squares solution.

8. **[3pt]** We believe that a real number Y is approximately determined by X by an unknown cubic polynomial

$$Y = aX^3 + bX^2 + cX + d .$$

We are given the following table of data connecting the values of X and Y:

X	0.0	0.5	1.0	1.5	2.0	2.5
Y	0.0	0.20	0.27	0.30	0.32	0.33

Using the above data points, write down five equations in the four unknowns a, b, c, d. The least squares solution to this system is known as the cubic polynomial of best fit. Write down the normal equations for this system, solve them in MATLAB and determine the cubic of best fit. Please also hand in your code.