Question:	1	2	3	4	5	6	7	8	Total
Points:	10	10	10	20	10	10	20	10	100

Answer the questions in the spaces provided. If you need more space for your answer, continue on the back of the page. Show all your work and indicate your reasoning in order to receive the credit. Present your answers in *low-entropy* form.

1. (10 points) Estimate the number of operations that are required to inverse $n \times n$ matrix using the method from HW05:

The algorithm in HW5 requires one LU decomposition and $2 \times n$ solutions of triangular systems of linear equations.

LU decomposition — $O(n^3)$ operations

solution of triangular sys. — $O(n^2)$ operations

Total count: $O(n^3) + 2 \cdot n \cdot O(n^2) \to O(n^3)$

2. (10 points) Estimate the number of operations that are required to calculate the determinant of an $n \times n$ matrix using the method from HW05:

The algorith in HW requires one LU decomposition and one product of $n+1$ factors.

LU decomposition — $O(n^3)$ operations

Product — $O(n)$ operations

Total count: $O(n^3) + O(n) \to O(n^3)$

3. (10 points) Estimate the number of operations that are required to calculate a polynomial of the order n using Horner's method. Compare with the number of operations for worst case "naive" method of calculations. For the purpose of this assignment assume that calculation of

x^n requires ~~n~~ n-1 multiplications. $P_n(x) = a_0 + a_1 x + a_2 x^2 + \dots a_n x^n$

Horner's method: $P_n(x) = a_0 + x \cdot (a_1 + x \cdot (a_2 + \dots + x \cdot (a_{n-1} + a_n x)$

n multiplications and n additions

Total count: $O(n)$

Naive: n additions — $O(n)$

$0 + 1 + 2 + 3 + n = \frac{n(n+1)}{2} \sim \frac{n^2}{2} \to O(n^2)$ multiplications

Total count: $O(n) + O(n^2) \to O(n^2)$

4. You are solving a nonlinear equation $f(x) = 0$ using **bisection** and starting from the initial interval $[a, b]$, $a = 1$ and $b = 2$. You are working in *quadruple precision*, that is you are working with 128 bit floating point numbers that have 112 bits in the fractional part and 15 bits in the exponent.

   (a) (10 points) estimate the best precision that you can achieve, i.e. the smallest length of the interval containing the solution of your equation. Express your answer as a power of 2.

   The best precision is of the order of the distance between adjacent floating point numbers. For the [1,2] interval this is just machine epsilon. For a 112 bit fractional part: the machine epsilon is $\varepsilon = 2^{-112}$

   (b) (10 points) estimate the number of function evaluations that is required to achieve the best possible precision.

   One starts with two function evaluation, f(1) and f(2). Each subsequent evaluation leads to the reduction of the interval by factor 2. It take ~112 iterations to reduce the internal from ~1 to ~2^{112}. So, there are ~$2 + 112 = 114 \sim 10^2$ function evaluations

5. (10 points) A hypothetical algorithm for solution of nonlinear equation achieves the cubic convergence rate. During your testing you noticed that after the three initial iterations the

error of the solution was 10^{-2}. How many more iterations do you need to reduce the error to better than 10^{-14}. Explain.

For the cubic convergence rate $\delta_n \sim (\delta_{n-1})^3$, where δ_n is the error after n iterations. Hence, $\delta_3 \sim 10^{-2} \to \delta_4 \sim 10^{-6} \to \delta_5 \sim 10^{-18}$, i.e. one needs two more iterations of the algorithm.

6. (10 points) Determine the interpolating polynomial, $P(x)$, that passes through the points $(0,0)$, $(\frac{1}{2}, \cancel{\frac{3}{2}}\,2)$, $(1, \cancel{1}\,\frac{1}{2})$. Construct the Lagrange polynomials $L_i(x)$ and write down $P(x)$.

$x_1=0, x_2=\frac{1}{2}, x_3=1$; $y_1=0, y_2=2, y_3=\frac{1}{2}$;

$$L_1(x) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} = \frac{(x-\frac{1}{2})(x-1)}{(-\frac{1}{2})(-1)} = \underline{2(x-\tfrac{1}{2})(x-1)};$$

$$L_2(x) = \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} = \frac{x(x-1)}{(-\frac{1}{2})(\frac{1}{2})} = \underline{-4x(x-1)};$$

$$L_3(x) = \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} = \frac{x(x-\frac{1}{2})}{1\cdot\frac{1}{2}} = \underline{2x(x-\tfrac{1}{2})};$$

$$P(x) = y_1\cdot L_1(x) + y_2\cdot L_2(x) + y_3\cdot L_3(x) = -8x(x-1) + \frac{1}{2}\cdot 2x(x-\tfrac{1}{2}) =$$
$$= x(-8x+8+x-\tfrac{1}{2}) = \underline{-7x^2+\tfrac{15}{2}x};$$

Matlab

7. The Airy function (or Airy function of the first kind), $Ai(x)$, is a special function named after the British astronomer George Airy (1801–1892). The function Ai(x) is a solutions to the differential equation

$$\frac{d^2y}{dx^2} - xy = 0.$$

The Airy function provides uniform semiclassical approximations near a turning point of Schrdinger's equation; the Airy function describes the intensity near an optical directional caustic, such as that of the rainbow; the Airy function is also important in microscopy and astronomy: it describes the pattern, due to diffraction and interference, produced by a point source of light (the one which is much smaller than the resolution limit of a microscope or telescope).

Matlab's function `airy(x)` returns the value(s) $Ai(x)$.

(a) (5 points) Plot a graph of Airy function for $-10 \le x \le 0$

(b) (10 points) Using the matlab code that was developed in class for the bisection, the secant, and the dekker's method, find the locations of two first negative zeros of Airy function.

(c) (5 points) Compare the performance of the methods. Describe your observations in gitlab's README file.

Git and Gitlab

8. (10 points) Upload the code you wrote for this exam:

   1. Use gitlab web interface to create a new project called **midterm2-sample** (the name is case sensitive, must be exactly as shown)
   2. Use gitlab web interface to create *README.md* file
   3. Use gitlab web interface to upload your matlab code to your project
   4. Use gitlab web interface to grant the access to your project (with the permission of the *reporter*) to the user `michael.rozman`