Please write your family and given names and **underline** your family name on the front page of your paper.
Submit at BA4226.

General note: Plotting quantity y versus quantity x, means that x is in the x-axis and y is on the y-axis, i.e. what follows "versus" is in the horizontal axis.

1. Consider computing the smallest root of the function $f(x) \equiv xe^{-\frac{x}{2}} - \frac{e^{-1}}{2}$, defined in $\mathbb{R}$.
(Note that $e = \exp(1) = 2.7183\ldots$ and $e^{-1} = \exp(-1) = \frac{1}{e} = 0.3679\ldots$)

(a) [*4 points*] Using mathematical arguments, show that there exists exactly one root of $f(x)$ in $[0, 2]$.
Using mathematical arguments, show that there exists exactly one root of $f(x)$ in $(-\infty, 6]$ (i.e. there are no roots in $(-\infty, 0)$ and no roots in $(2, 6]$). You can use the fact that $f(6) > 0$.
Let r denote the root of $f(x)$ in $(-\infty, 6]$.

(b) [*4 points*] Using $x^{(0)} = 0$ as initial guess, apply (by hand) one Newton iteration to compute an approximation $x^{(1)}$ to the root. Indicating how $x^{(1)}$ is computed, simplify as much as you can, and write the result in terms of e^{-1}.

(c) [*4 points*] The equation $f(x) = 0$ can be equivalently written as $x = \frac{e^{-1}}{2} e^{\frac{x}{2}}$, which gives rise to the fixed-point iteration scheme $x^{(k+1)} = g(x^{(k)})$, with $g(x) = \frac{e^{-1}}{2} e^{\frac{x}{2}}$.
Using $x^{(0)} = 0$ as initial guess, apply (by hand) one fixed-point iteration to compute an approximation $x^{(1)}$ to the root. Indicating how $x^{(1)}$ is computed, simplify as much as you can, and write the result in terms of e^{-1}.

(d) [*10 points*] Show that the fixed-point iteration scheme converges to r, if started anywhere in $[0, 2]$.

(e) [*4 points*] What is the convergence rate (order) of Newton's if started close enough to r? Explain.
What is the convergence rate (order) of the fixed-point iteration in (c)-(d)? Explain.

(f) [*10 points*] Find the largest interval you can, so that the fixed-point iteration scheme converges, if started within the interval, and so that you justify your answer mathematically. (Note: the interval may expand $[0, 2]$ from both sides.)

(g) [*14 points*] Using MATLAB, implement the above fixed-point iteration, as well as secant and Newton's methods and apply them to solve $f(x) = 0$. (For Newton's method, you can define $f'(x)$ explicitly; no need to use any symbolic package.) Run the following experiments:
− apply the fixed-point iteration method with initial guess 0
− apply the secant method with initial guesses 0 and 2
− apply Newton's method with initial guess 0.
As stopping criterion, for all the Newton's and secant experiments use $|f(x^{(k)})| \leq 10^{-10}$, and for the fixed-point iteration experiments use $|g(x^{(k)}) - x^{(k)}| \leq 10^{-10}$. For each experiment and for each iteration, output the iteration index, the current approximation to the root, and the current residual (Newton's, secant) or the current difference $|g(x^{(k)}) - x^{(k)}|$ (fixed-point). Use appropriate format to show the details of the results (e.g. `fprintf('%3d %15.12f %10.2e\n', .. );`). In one graph for all the experiments, and in log scale for the y variable (`semilogy`), plot the absolute value of the error of the respective approximation versus the iteration index. Distinguish each line plotted from the rest, e.g. fixed-point solid line, secant dashed line, Newton's dotted line. When computing the error, use r1 = 0.203656862188284 as the exact value of the (smallest) root. Discuss what you observe.
Hand-in a hard-copy of your code, output and plot and your hand-written or typed comments.

2. Consider the system of two nonlinear equations with respect to the two unknowns x and y

$$4x^2 + y^2 = 25$$
$$xye^{-\frac{x+y}{4}} = 0.2$$

(a) [*3 points*] Formulate the Jacobian matrix for the above nonlinear system. The entries of the Jacobian should be given in terms of the variables x and y.

(b) [*3 points*] Assume one wants to apply Newton's method to the above system with starting guess $[x^{(0)}, y^{(0)}]^T$. Find conditions on $x^{(0)}$ and $y^{(0)}$, so that Newton's method is applicable and explain. (Simplify the condition(s) as much as you can.) Is Newton's method applicable to the above system with starting guess $[1, 1]^T$? Explain.

(c) [*4 points*] Apply (by hand) one Newton iteration to the above system with starting guess $[1, 0]^T$. Show your calculations.

3.

(a) [*5 points*] Using the Newton's Divided Differences (NDD) table, construct the polynomial $p_2(x)$ of degree at most 2, that interpolates the function $f(x) = e^x$, at the points $x_0 = -1$, $x_1 = 0$, $x_2 = 1$. Bring the polynomial into monomial form, if it is not already in that form. The coefficients of the polynomial are to be given either in terms of e and e^{-1}, or as (approximate) numerical values.

(b) [*3 points*] Give the error formula for this interpolation problem (i.e. for *this* f and *these* data points). The formula should involve an unknown point ξ. Any other functions involved in the formula should be given explicitly in terms of x.

(c) [*12 points*] Using the polynomial interpolation error formula, give an upper bound for each of $|e^{1.5} - p_2(1.5)|$, $\max_{-1 \le x \le 1} |e^x - p_2(x)|$, $\max_{-1 \le x \le 2} |e^x - p_2(x)|$, $\max_{-2 \le x \le 1} |e^x - p_2(x)|$. Explain how you got the bounds. The bounds are to be given as (approximate) numerical values.
Note: You must use the polynomial interpolation error formula to find the bounds.

4. [*20 points*] Consider the set of data $\{(x_i, y_i), i = 0, \cdots, 20\}$ given by the following piece of MATLAB code:

```
x = [ 0.9  1.3  1.9  2.1  2.6  3.0  3.9  4.4  4.7  5.0  6.0  7.0  8.0  9.2 ...
     10.5 11.3 11.6 12.0 12.6 13.0 13.3];
y = [ 1.3  1.5  1.85 2.1  2.6  2.7  2.4  2.15 2.05 2.1  2.25 2.3  2.25 1.95 ...
      1.4  0.9  0.7  0.6  0.5  0.4  0.25];
```

Consider also the set of equidistant evaluation points $\{(xi)_i, i = 1, \cdots, 100\}$ in the interval [0.87, 13.33]. Write a MATLAB program that

- constructs the not-a-knot cubic spline interpolant of the above data and evaluates it on the evaluation points
- constructs the 20th degree polynomial interpolant of the above data and evaluates it on the evaluation points
- plots the above two interpolants in one graph based on their respective values on the evaluation points.

Comment on what you observe, including any warning obtained by MATLAB.

Notes:

- This question is inspired by the relevant discussion in pages 150−153 of the Burden and Faires textbook.
- You can find an electronic version of the above piece of MATLAB code in `http://www.cs.toronto.edu/~ccc/Courses/336/duck.m`. (Hence, no excuses for typographical errors...)
- Use `yvs = interp1(x, y, xv, 'spline');` (or `yvs = spline(x, y, xv);`) to get the values of the cubic spline interpolant on the evaluation points. This interpolant uses a type of not-a-knot condition.
  Use the built-in MATLAB functions `polyfit` and `polyval` to get the values of the 20th degree polynomial interpolant on the evaluation points. Note that although the MATLAB built-in function `polyfit` is designed to construct least-squares polynomial approximations, it can be used to construct polynomial interpolants as well, if the number of data and the degree requested are set appropriately.
  If you get a warning (Warning: Polynomial is badly conditioned...") from `polyfit`, you are not necessarily doing anything wrong.
- Use `linspace(0.87, 13.33, 100)` to get the 100 equidistant evaluation points.
- Set the axes scales using `axis([0 14 -6 5])` to view the plot appropriately. (Use `axis` *after* `plot`.)
- Use solid line for the spline interpolant and dashed for the polynomial one. For each data point, write an `o` on the graph.
- Hand in a hard-copy of your code and plots, as well as your comments.