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

1. Let $f(x) = e^x - 4x^2$.

(a) [*3 points*] Show that f has (at least) one root in [0, 1].

(b) [*2 points*] Let $g(x) = \frac{1}{2} e^{\frac{x}{2}}$. Verify that $g(x) = x$, iff $f(x) = 0$.

(c) [*15 points*] Show that g has a unique fixed point in [0, 1], and, therefore, that $f(x)$ has a unique root in [0, 1].
Show that the fixed-point iteration $x^{(k+1)} = g(x^{(k)})$, with $g(x) = \frac{1}{2} e^{\frac{x}{2}}$, converges to the unique root of $f(x)$ in [0, 1] for any starting guess chosen in [0, 1].

(d) [*5 points*] What is the rate of convergence of the fixed-point iteration with $g(x) = \frac{1}{2} e^{\frac{x}{2}}$? Explain.

(e) [*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, 1] from both sides.)

Help: First expand [0, 1] to the left, then to the right, as much as you can.

2. [*20 points*] Consider a robot arm with two joints (as in the notes). The first arm part starts at (0, 0), ends at (x_m, y_m), and has length a. The second arm part starts at (x_m, y_m), ends at (x_p, y_p) (tip of the arm), and has length b. The angle of the first arm part with the x-axis is θ, and that of the second arm part with the first is ϕ. As in the notes, we have the equations

$$x_p = a\cos(\theta) + b\cos(\theta + \phi) \tag{1}$$

$$y_p = a\sin(\theta) + b\sin(\theta + \phi) \tag{2}$$

Given a, b, x_p and y_p, equations (1)-(2) form a 2×2 nonlinear system of equations, with respect to θ and ϕ.
Give (by hand) the Jacobian $J(\theta, \phi)$ for this system.
Write a MATLAB function that implements Newton's method for the system of equations (1)-(2). (The code only needs to be for (1)-(2), and not for a general 2×2 system of equations.) Use as stopping criterion the absolute (Euclidean) norm of the residual. Write also a MATLAB script that, given values to `a, b, xp` and `yp`, uses the above function to solve the above 2×2 system of nonlinear equations, then, using the values of θ and ϕ computed, plots the arm (draws a picture of the arm) at the final location using solid line. In the same plot, plot the initial location of the arm using dashed line.
Run your script for `a = 2; b = 1; xp = 1.4; yp = 1.7;`
Do this twice, once with initial guess $[\frac{\pi}{4}, -\frac{\pi}{2}]^T$, and once with initial guess $[\frac{\pi}{4}, \frac{\pi}{2}]^T$.
The script should output the two angles calculated and the Euclidean norm of the residual for each iteration. Use tolerance 10^{-8}. Hand-in a hard-copy of your code (script and function), the output and the plots.

Notes:

The function that implements Newton's method for (1)-(2) should start with

```
function [r, it, rall, res] = robotarm(a, b, p, r, tol)
```

The point where we want to position the robot arm is in the 2×1 vector p. On input, r is the vector of the two initial angles, while on return, r is a vector of the two (final) angles calculated. Moreover, *it* is the number of Newton's iterations, *rall* an $(it + 1) \times 2$ array of pairs of approximate angles calculated at each iteration, including the initial pair of angles, and *res* a vector of the Euclidean norms of the residuals at each iteration.

To compute the vector $v = J^{-1}(\bar{x}^{(k)})\bar{f}(\bar{x}^{(k)})$, you need to **solve** the linear system $J^{-1}(\bar{x}^{(k)})v = \bar{f}(\bar{x}^{(k)})$, using backslash. Note that $f_1(\theta, \phi) = a\cos(\theta) + b\cos(\theta + \phi) - x_p$, and $f_2(\theta, \phi) = a\sin(\theta) + b\sin(\theta + \phi) - y_p$.

Once you have the approximate solution (angles) calculated, you need to calculate x_m and y_m to plot the arms.

Use `axis equal` *after* you plot the robot arm, for uniformity and for more appropriate aspect ratio of the axes.

3. Consider the function $f(x) = x \ln x$, and the data $(e^{-1}, -e^{-1})$, $(1, 0)$ and (e, e) arising from f.
(Note: $e = 2.7183\ldots$ and $e^{-1} = 0.3679\ldots$, but it is easier for you to maintain the symbolic values e and e^{-1}.)

(a) [*5 points*] Using Newton's Divided Differences (NDD), give the quadratic polynomial $p_2(x)$ interpolating $f(x)$ at the above data. The coefficients of the polynomial are to be given in terms of e and e^{-1}.

You do **not** have to bring $p_2(x)$ into monomial form.

(b) [*3 points*] Give the error formula for this interpolation problem (i.e. for this f and these data points). The formula should involve only an unknown point ξ, the values e^{-1} and e, as well as a few constants.

(c) [*6 points*] Assume we evaluate p_2 at some $x \in [e^{-1}, 3]$. Indicate the interval where ξ belongs to. Find an upper bound (as sharp as you can) of the error $|f(x) - p_2(x)|$ when $x \in [e^{-1}, 3]$, and explain how your got it. The bound is to be given in terms of e^{-1} and e. You may specify it as the maximum of a fixed number of quantities written in terms of e^{-1} and e, and simplified as much as possible. If the quantity $t = e^{-1} + 1 + e$ appears multiple times, you may denote it by t, and work out some results in terms of t.
Then, find an approximate numerical value for the bound.

(d) [*6 points*] Assume we evaluate p_2 at some $x \in [e^{-2}, e]$. Indicate the interval where ξ belongs to. Find an upper bound (as sharp as you can) of the error $|f(x) - p_2(x)|$ when $x \in [e^{-2}, e]$, and explain how your got it. The bound is to be given in terms of e^{-2}, e^{-1} and e. You may specify it as the maximum of a fixed number of quantities written in terms of e^{-2}, e^{-1} and e, and simplified as much as possible. If the quantity $t = e^{-1} + 1 + e$ appears multiple times, you may denote it by t, and work out some results in terms of t.
Then, find an approximate numerical value for the bound.

4. Consider the piecewise cubic polynomial

$$S(x) = \begin{cases} S_0(x) = a_0 + a_1(x+1) + a_2(x+1)^2 + a_3(x+1)^3 & \text{for} \quad -1 \le x \le 0 \\ S_1(x) = b_0 + b_1(x-1) + b_2(x-1)^2 + b_3(x-1)^3 & \text{for} \quad 0 \le x \le 1 \end{cases} \tag{1}$$

defined in $[-1, 1]$ with respect to the knots $x_0 = -1$, $x_1 = 0$ and $x_2 = 1$.

(a) [*7 points*] Write all conditions that a_i and b_i, $i = 0, \cdots, 3$, have to satisfy so that $S(x)$ is a cubic spline. Explain what each condition means. How many conditions have you written?

(b) [*8 points*] Write all additional conditions that a_i and b_i, $i = 0, \cdots, 3$, have to satisfy so that $S(x)$ is the clamped cubic spline interpolant of $f(x) = \sin(\pi x)$. Explain what each condition means. How many additional conditions have you written?
Note: You don't need to solve the equations, and you don't need to give the values of the parameters.
Recall that the clamped end-conditions involve the first derivative of f.

(c) [*10 points*] Consider using three types of interpolation for the function $f(x) = \sin(\pi x)$, namely, polynomial, linear spline and cubic spline interpolation, as in script `http://www.cs.toronto.edu/~ccc/Courses/336/scriptint.m`
Add the appropriate extra lines to the script to compute the *clamped* cubic spline interpolant at the points indicated. (Use `spline(xi, yi2, xe);` where `yi2` has two extra values.) Then run the script and collect the output. (If you get a warning from `polyfit`, you are not necessarily doing anything wrong.) Comment on the results, in particular, how the error of each interpolant behaves as the number of data points increases, and whether this behaviour agrees with what expected from theory.
Do not change the format according to which the errors are output. However, you need to output a few more quantities for the linear and cubic spline interpolants, more specifically, the experimentally observed orders of convergence for pairs of n's, in order to better justify your comments about the error behaviour. Submit a hard-copy of your completed script, the output, and the comments.