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

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 the function $f(x) = \cos(x) - \frac{x^2}{2} + \frac{1}{2}$.

(a) [*2 points*] Using appropriate graphs, locate approximately the positive root of $f(x)$. (Choose functions easy to graph, so that you can draw the graphs by hand.)

(b) [*3 points*] Using mathematical arguments, show that there exists exactly one positive root of $f(x)$, and indicate an interval $[a, b]$ of length not greater than $\frac{\pi}{4}$, where the positive root lies.

(c) [*5 points*] Using $x^{(0)} = \frac{\pi}{2}$ 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, write the result in terms of π, then give a numerical value.

(d) [*15 points*] Consider the equation $f(x) = 0$ written as $x = \sqrt{2\cos(x) + 1}$, and the associated fixed-point iteration method, with $\phi(x) = \sqrt{2\cos(x) + 1}$.
Apply (by hand) one fixed-point iteration with initial guess $x^{(0)} = \frac{\pi}{2}$, and indicate the value of $x^{(1)}$.
(Note: this $x^{(1)}$ is different than the one in (c).)
Let $\delta = 0.01$. Show that if $x^{(0)} \in [\frac{\pi}{4}, \frac{\pi}{2} - \delta]$, the fixed-point iteration converges to the positive root of f.
What is the order of convergence of the fixed-point iteration? Explain.

(e) [*5 points*] In `http://www.cs.toronto.edu/~ccc/Courses/336/bisect.m` you are given some MATLAB code that implements the bisection method. Write the code that implements Newton's method (for a given scalar function f), and the code that implements the general fixed-point iteration $x^{(k+1)} = \phi(x^{(k)})$, for a given $\phi(x)$. The MATLAB function for Newton's should start with `function [r, it, rall, res] = newton(fun, a, tol, maxit)` and the MATLAB function for fixed-point iteration should start with `function [r, it, rall, diff] = fixedpt(fun, a, tol, maxit)` The meaning of the input and output arguments should be obvious, after you study the given code for bisection. (You do not need to return a range around the root for Newton's or fixed-point iteration. The output variable `diff` is the vector of differences $\phi(x^{(k)}) - x^{(k)}$, which you can view as the ''residual'' of the fixed-point iteration.) For Newton's, use as stopping criterion the absolute norm (value) of the residual (of $f(x)$). For fixed-point iteration, use as stopping criterion the absolute norm (value) of the residual of $\phi(x)$. For Newton's, you can hard-code f and f' as in `http://www.cs.toronto.edu/~ccc/Courses/336/ff.m`, and for fixed-point iteration, you can hard-code ϕ in a similar way. Hand-in a hard-copy of your codes. Also, submit your codes through the cdf website.

(f) [*10 points*] Consider the function $f(x) = \cos(x) - \frac{x^2}{2} + \frac{1}{2}$, and the associated fixed-point iteration with $\phi(x) = \sqrt{2\cos(x) + 1}$. Consider also the script in `http://www.cs.toronto.edu/~ccc/Courses/336/scriptnl.m` Run the script and hand-in a hard-copy of its output. Comment on the results, especially on the rate of convergence of the iterative methods, and the number of iterations taken to converge. Do they agree (approximately) with what expected from the theory? What can you say about the asymptotic error constant of the fixed-point iteration above?
Note: The exact root to this function is not known, but you can get a good approximation by MATLAB's `fzero` function which gives `root = 1.265423705869335`. You can take that as ''exact''.
Do not alter the format of output. Use tolerance 10^{-7}.

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

$$x_p = a\cos(\theta) + b\cos(\theta + \phi) \quad (1)$$

$$y_p = a\sin(\theta) + b\sin(\theta + \phi) \quad (2)$$

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 and the initial values given, plots the arm (draws a picture of the arm) at the final position by solid lines, and at the initial position by dashed lines.
Run your script for `a = 2; b = 1; xp = 1; yp = 1;`
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, using `format long`, the two angles calculated and the Euclidean norm of the residual for all iterations. Use tolerance 10^{-8}. After each plot, use `axis([-0.05 2.15 0 2]);`, to keep a consistent look for the plots. Hand-in a hard-copy of your code (script and function), the output and the plots. Also, submit your codes through the cdf website.

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, maxit)`
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 approximate angles calculated in all iterations including the initial 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 $J(\bar{x}^{(k)})v = \bar{f}(\bar{x}^{(k)})$, using backslash.

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

3. Consider the function $f(x) = \sqrt{x}$, and the data $(\frac{1}{4}, \frac{1}{2})$, $(1, 1)$ and $(4, 2)$ arising from f.

(a) [*10 points*] Using the NDD basis functions and the associated NDD table, construct the quadratic polynomial $p_2(x)$ interpolating $f(x)$ at the above data.

(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) [*10 points*] Assume we evaluate p_2 at some $x\in[\frac{1}{16}, 4]$. Indicate the interval where ξ belongs to and explain. Find an upper bound (as sharp as you can) for the error $|f(x) - p_2(x)|$ when $x\in[\frac{1}{16}, 4]$. Give a numerical value to the bound.

(d) [*7 points*] Assume we evaluate p_2 at $x = \frac{1}{16}$. Indicate the interval where ξ belongs to and explain. Find an upper bound (as sharp as you can) for the error $|f(x) - p_2(x)|$ when $x = \frac{1}{16}$. Give a numerical value to the bound. Then, calculate (by hand or computer) the actual value of $|f(x) - p_2(x)|$ when $x = \frac{1}{16}$, and comment.

Note: All parts of this question are to be done by hand, except possibly the calculation $|f(x) - p_2(x)|$ when $x = \frac{1}{16}$.

4. [*10 points*] Consider using three types of interpolation for the function $f(x) = \sqrt{x}$, 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 cubic spline interpolant at the points indicated. 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 and other quantities are output. You need to explain what the other quantities represent. Submit a hard-copy of your completed script, the output, and the comments. Also, submit your code through the cdf website.