Work through each of the following problems. Be sure to follow the midterm project guidelines found in the resources section on sakai. Turn in all coding portions digitally and the all analysis portions in class. Both portions must be turned in by the due date and time.

1. (3 points) **Implementing a MATLAB root finder**

   For this project you must implement a root finder function called `findroot` based on Newton's method. It will be utilized in the remainder of this project (and throughout the course). All applications requiring root finding/equation solving should call this function only. Your implementation should be capable of (at least) the following calls:

   findroot(f, f', x_0) $\leftarrow x$ attempts to return x satisfying $|f(x)| < 10^{-10}$ by Newton iteration with initial guess, x_0, using no more than $N = 100$ iterations. Otherwise, returns $x =$ NaN.

   findroot(f, f', x_0) $\leftarrow$ `x` optional output returns a vector containing all of the Newton iterates.

   findroot$(f, f', x_0, \epsilon, N)$ $\leftarrow x$ optional input to specify the maximum number of iterates and maximum defect threshold.

2. (8 points) **Adventures in root finding**

   Use `findroot` to answer the following questions. You may also use the builtin functions `sin`, `cos`, `exp`, `rand`, `polyval`, and `deconv` if needed.

   (a) Write a script called `local_extrema` Find all local extrema for the function $f(x) = \exp(-t\cos(\pi t))$ for $t \in [0, 4]$. Use MATLAB's plotting features to choose initial guesses for `findroot` and to verify that you have found them all.

   (b) Consider the equation $\sin x = x$ which has a unique solution at $x = 0$. Write a script called `sin_fp` which uses `findroot` to solve this equation with varying initial points, number of iterations, and defect tolerances. Does the Newton iteration appear to be converging? If so, estimate the order of convergence. If not, explain why it does not converge.

   (c) The function $g(x) = 4x - 4x^2$ has exactly four 2-periodic points on the interval $[0, 1]$ (i.e. solutions to the equation $g(g(x)) = x$). Write a MATLAB script named `periodic_demo` which attempts to find these points by doing calls to `findroot` using initial data generated randomly in $[.08, 1]$ until it successfully returns a solution 1,000 times. Count the number of trials which converged to each of the four solutions (Be careful when implementing the equality check).

   (d) Did the script in part (b) find all the roots equally often? Did it even find them all once? Analyze and explain why this count is so skewed (Hint: see Theorems 2.4 and 2.6). Write a new script named `periodic_demo_v2` which fixes the problem. The new version should continue choosing initial data randomly in $[0.8, 1]$ but it should only run until it finds all four solutions at least once. It should reliably do this in less than 20 (and probably a lot fewer) calls to `findroot`.

Truncation error analysis

3. (8 points) Recall that the exponential function is defined by the following power series

$$\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{1}{2}x^2 + \mathcal{O}(x^3) \text{ as } x \to 0.$$

   (a) Show that for any $x \geq 0$, the second order Taylor polynomial remainder satisfies the inequality

$$|R(x)| = \left| 1 + x + \frac{1}{2}x^2 - \exp(x) \right| \leq \frac{3^x x^3}{6}.$$

   (b) Use this result to write a MATLAB script named `exp_demo` which approximates the value of $\exp(\frac{1}{2})$. Compare this result with the builtin function `exp` and verify your truncation error satisfies the bound from (a).

   (c) Find the largest value, $b > 0$, such that the approximation error from (a) is bounded by 10^{-2} for all $x \in [0, b]$ i.e. the following error bound holds

$$|R(x)| \leq \frac{3^x x^3}{6} \leq 10^{-2}.$$

   (Hint: Turn it into a root finding problem.)

   (d) Write a function named `expsolver` which uses a 10^{th} order Taylor approximation and calls to `findroot` to compute $\exp(x)$. You do not have to analyze the truncation error in this case. Compare your implementation with the builtin function `exp` for evaluating $x = 3$ and $x = 1000$. Are the differences comparable? If not, explain why.

Newton vs quadratic formula

4. (6 points) Consider the quadratic polynomial

$$p(x) = 0.01x^2 + 1000x + 0.01.$$

   (a) Use `findroot` to construct a new function `quadraticroots` with the following call.

   quadraticroots$(a, b, c) \leftarrow (r_1, r_2)$ returns both roots of the polynomial $ax^2 + bx + c$.

   This function should call `findroot` to find **both** roots (i.e. do not simply find one root and divide it out).

   (b) Create a MATLAB script named `quad_stability` which computes the roots of p using the following methods:

   1. The builtin function `roots`.
   2. The function `quadraticroots` implemented above.
   3. The classic quadratic formula.

   Assuming that the `roots` function is very accurate and numerically stable, use the roots it produces to compute the absolute and relative error for the roots found using the other two methods. Verify that the solutions produced by `quadraticroots` are also highly accurate.

   (c) What about the roots obtained from the quadratic formula? Analyze this case and explain why the error is so large despite the fact that the formula gives an exact solution.