Assignment 4

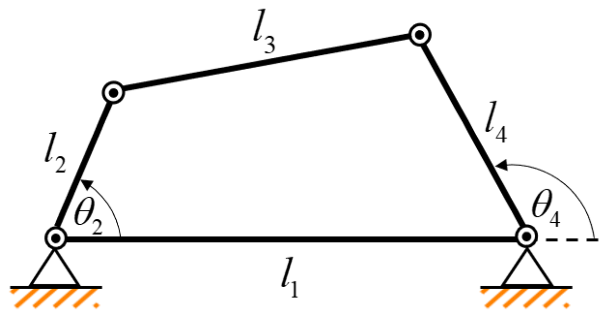
Due: Thursday November 5, 2020 at 11:59pm
MAE 107 Computational Methods for Engineers
Professor Michael Tolley

Homework Submission

- Homework submission is through Gradescope. Please leave extra time for submission, especially if you have not used Gradescope previously.
- Assignments are graded based on correctness, how well you organize your homework (i.e. it should be easy to understand your thinking and easy to find your responses), and how well you follow the submission instructions below.
- You will not receive credit if you just give an answer. Your solution must demonstrate how you got the answer.
- If you ever think a problem is stated incorrectly, not enough information is given, or it is impossible to solve, don't panic! Simply make a reasonable assumption that will allow you to solve the problem (but clearly state what this assumption is), or indicate why it is not possible to solve the problem.
- Please attend office hours (listed in the syllabus) if you have any questions regarding the assignment.
- You are welcome to discuss the assignment with other members of the class, but everything you submit should be your own (i.e. you wrote it, typed it, or generated the plot in Matlab).

Problem 1 (40 points): Root Finding, Fixed Point Iteration

The following figure shows a four-bar linkage, used in many mechanisms including windshield wipers, rear mountain bike shocks, etc.



This is a one degree-of-freedom mechanism (only one number is required to fully specify the configuration of the entire mechanism). For example, if θ_2—the angle of the second link with length l_2—is considered the input (e.g. from a motor), then θ_4—the angle of the fourth link with length l_4—could be the output (e.g. the position of the windshield wiper blade). With kinematic analysis, the angles θ_2 and θ_4 can be related with *Freudenstein's Equation*:

$$-\rho_1 \cos\theta_2 + \rho_2 \cos\theta_4 + \rho_3 = \cos(\theta_2 - \theta_4)$$

where

$$\rho_1 = \frac{l_1}{l_4}, \rho_2 = \frac{l_1}{l_2}, \rho_3 = \frac{l_1^2 + l_2^2 + l_4^2 - l_3^2}{2 l_2 l_4}$$

a) Solve Freudenstein's Equation to put it in the form $\theta_4 = g(\theta_4)$ (Hint: the right hand side of this equation should have an inverse cosine)

b) Using the equation from a), write a Matlab implementation of the Fixed Point Iteration algorithm. Use the following link lengths:

- $l_1 = 7.0$
- $l_2 = 1.8$
- $l_3 = 4.2$
- $l_4 = 4.8$

Use your algorithm to find the value of θ_4 corresponding to a $\theta_2 = 45°$ to within $\epsilon_a = 0.1°$. Start with an initial guess of $\theta_4 = 100°$. Report the number of iterations required. (Note, if you get an imaginary result, it means the value inside the inverse cosine is out of the range [-1,1] which could be due to a typo or a bad initial guess.)

c) Plot the four-bar linkage in the current position. To do this, it may help to note that the positions of the four rotational joints (clockwise from the left) are given by:

- $p_1 = (0, 0)$
- $p_2 = (l_2 \cos\theta_2, l_2 \sin\theta_2)$
- $p_3 = (l_1 + l_4 \cos\theta_4, l_4 \sin\theta_4)$
- $p_4 = (l_1, 0)$

d) Finally, wrap a *for* loop around your entire code to plot the position of the four-bar linkage as you adjust θ_2 from $0°$ to $360°$ in 10 degree increments. You should use the root from the previous angle as your initial guess for the new root (bonus question: what happens if you don't?). Use the command *hold on* to plot all the configurations on the same plot. What path does the follower link trace? Does it make a full rotation? (Note: If you want to make your plot look pretty, you can plot only the positions of the joints using *scatter(x,y)* and only plot the links for a single value of $theta_2$.)

Problem 2 (20 Points): Root Finding, Newton-Raphson Method

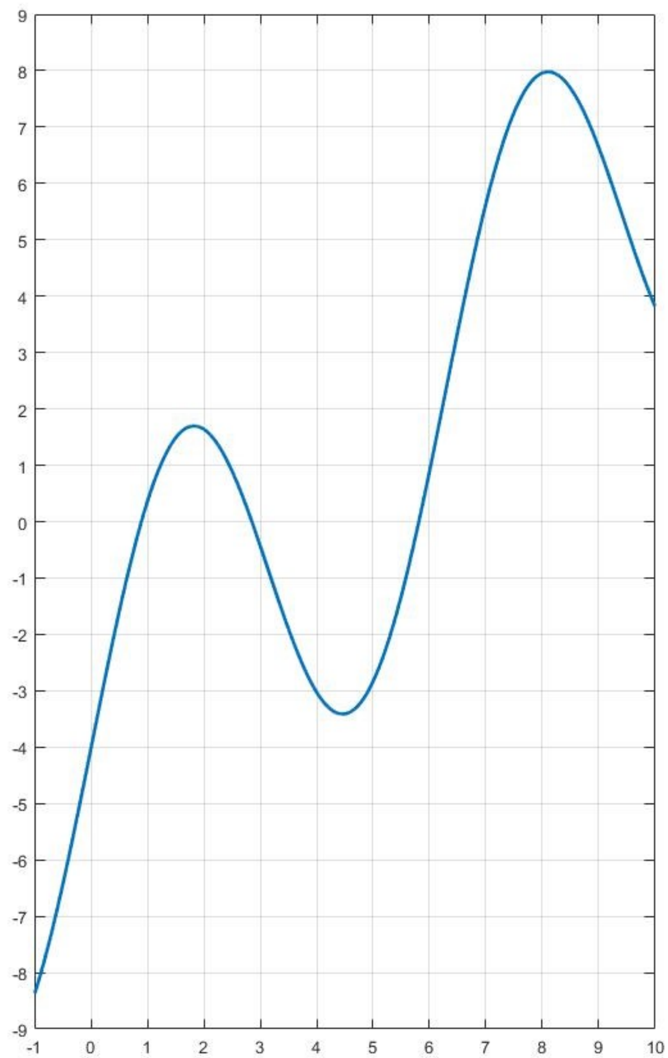
a) Re-write Freudenstein's Equation from Problem 1 in the format $f(\theta_4) = 0$.

b) Compute $f'(\theta_4)$.

c) In Matlab, write an iterative Newton-Raphson method to replace the one you used in Problem 1. Compute θ_4 for the parameters and settings specified in 1b), and compare the angle, error, and iterations with those found using Fixed-Point Iteration.

Problem 3 (20 points): 1D Optimization, Golden-Section Search

Graphically implement three iterations of a *Golden Section Search* on the following graph with initial $x_l = 0$ and $x_u = 9$. Each time you need to evaluate the function for the search, estimate the value of the function to two decimal places, and write the value on the graph. When you eliminate a portion of the graph from your search, shade it out to indicate the portion you eliminated. How many total function evaluations were required?



Problem 4 (20 points): 2D Optimization, Random Search

Implement a random 2D search to find the maximal value of the function:

$$f(x, y) = \sin(4x + 3y + 2x^2 - x^4 - 3xy - 2y^2)$$

In the range x = [-2,2], y = [-2,2]. The algorithm is very simple. Choose random values of x and y in the ranges (e.g. `x = -2 + 4*rand` in Matlab), then compute the corresponding value of $f(x, y)$. If it is higher than any previous value, store it at the new optimal value as well as the corresponding x and y coordinates. Use 1000 iterations. Report the values of x, y, and $f(x, y)$ of the maximum. Hint: To see if you're on the right track, you can plot the function in Matlab, or at `https://www.wolframalpha.com/` with the command:

```
plot sin(4x + 3y + 2x2  x4  3xy  2y2) x=-1..1, y=-1..1
```