Project 1: Kinematics of the Stewart Platform.
Due date: 10/26/2022

Statement of the Project

Solve the "Reality Check 1" project that is presented in the textbook *Numerical Analysis* by Timothy Sauer (3rd edition, pages 70 – 73). More specifically, **solve activities 1–7** (activity 8 is not necessary), with the following modifications:

- You don't need to use MATLAB. You can use jupyter/python instead. In this case, you may disregard all MATLAB hints in the text, such as

  ```
  You may use the @ symbol as described in Appendix B.5 to assign a function handle to
  your function file in the plotting command.  You may also need to precede arithmetic
  operations with the "." character to vectorize the operations, as explained in Appendix
  B.2.
  ```

- In activity 4, add the following questions:
  - Which equation solver did you use, and why? (e.g. bisection, FPI, ...)
  - How did you initialize your solver, and why? (e.g. how did you pick the initial interval for bisection, or initial condition for FPI, ...).
  - What stopping condition did you use in your solver? How accurate is the obtained solution (root)?
  - How fast is your solver? (e.g. linearly convergent, quadratically convergent, ...).

The deliverables consist of a Project Report and a Code Listing.

Project Report

This is a typed-up report (does not need to be long, 5 pages is okay) that carefully discusses and presents your **solutions to the project questions**. **A solution with no explanation may not receive full credit**.

If the activity asks for a numerical solution (e.g. activity 6), always include the numerical solution in the report. (The numerical solution may be a number, a vector, a list of numbers, a table, ...). Be sure to use enough significant digits!

If the activity asks for a plot (e.g. activity 3), always include the plot in the document.

For instance:

- In activity 1, explain how you tested function $f(\theta)$, and show the results of your tests.
- In activity 2, show the plot of $f(\theta)$, and explain how you have verified that there are roots at $\pm\pi/4$.
- In activity 4, plot $f(\theta)$. Use the equation solver to find all four poses **and list them** (i.e. list the resulting x, y, θ values for each pose). Explain how you tested your answers, and show the results of your tests. Be sure to discuss (guided by the questions above) your choice of equation solver, initial conditions, stopping conditions, accuracy and efficiency of the method.

Please explain any other important methodological decision that you made. For instance (these are just hypothetical examples), if you

- decided to try several methods and compare their performance;
- combined several methods into one super-method;
- tried several stopping conditions, and decided to use this one or another, because . . . ;
- timed the running time of your code and report it in a table;
- performed a fantabulous check to validate your results are correct;
- greatly optimized your code using such and such trick;
- were not able to solve this activity because you didn't know how to invert matrix M which is too large to store in memory.
- etc, etc, etc

The report should be submitted to the instructor **via Canvas as a PDF** file by the due date.

Code Listing

Your code should be clearly written, consistent with best practices, and **reproducible**. In particular:

- **Write your own numerical code**. The goal of this course is to learn numerical methods. You must program the **numerical methods explained in class** yourselves. Don't use readily-available numeric libraries, such as SciPy's `fsolve` or MATLAB's `vpasolve` to find roots of an equation. Don't use python's `solve_ivp` to solve an ODE. And so on.

  Of course, you may use libraries for all other things (e.g. for data manipulation, plotting, etc.). You may also use libraries for standard numerical computations (e.g. standard functions like trigonometric functions, matrix algebra, . . .)

- **Reproducibility**. Your code must be **ready to run** and **produce the same results** documented in the project report. Points will be deduced otherwise (e.g. I press "Run" and it hangs...).

  I recommend to add a brief README.txt file explaining how to run your code (e.g. "Run the jupyter notebooks activity1.ipynb, activity2.ipynb", ... Or: "First check that that package TDA.R is installed in the system. Then execute the command `Rcmd activity1.r` from the command line").

- **Structure your code**. Divide your code into files with meaningful names (e.g. activity1.m, activity2.m, . . .). Use **functions** to encapsulate relevant pieces of code. Use meaningful names for constants/variables (*don't* use "Dracula", "foo", "mystring", . . .). If you like, use classes / Object Oriented Programming (but it's not necessary for this course).

- **Comment your code**. Be as verbose as necessary, adding comments profusely whenever code is complex.

- **Test key functions**. If a function performs some complex task and is very important for your project, test it on a few known examples. The more complex the function, the more exhaustive the testing.

- **Reference your sources**. Do not “borrow” code from other people (websites, textbooks, research papers . . .) unless you cite them. No exceptions!

The code should be submitted to the instructor **via Canvas as a Zip (compressed)** file by the due date.