10. — PHY 494: Homework assignment (35 points total)

Due Thursday, April 4, 2019, 1:30pm.

Submission is to your **private GitHub repository**.

Read the following instructions carefully. Ask if anything is unclear.

1. `cd` into your assignment repository (change *YourGitHubUsername* to your GitHub username) and run the update script `./scripts/update.sh` (replace *YourGitHubUsername* with your GitHub username):

   ```
   cd  assignments-2019-YourGitHubUsername
   bash ./scripts/update.sh
   ```

   It should create three subdirectories[1] `assignment_10/Submission`, `assignment_10/Grade`, and `assignment_10/Work`.

2. You can try out code in the `assignment_10/Work` directory but you don't have to use it if you don't want to. Your grade with comments will appear in `assignment_10/Grade`.

3. Create your solution in `assignment_10/Submission`. Use Git to `git add` files and `git commit` changes. Files with output and code should be named exactly as required in the assignment.

4. When you are ready to submit your solution, do a final `git status` to check that you haven't forgotten anything, commit any uncommited changes, and `git push` to your GitHub repository. Check on *your* GitHub repository web page[2] that your files were properly submitted.

   You can push more updates up until the deadline. Changes after the deadline will not be taken into account for grading.

[1]If the script fails, file an issue in the Issue Tracker for PHY494-assignments-skeleton and just create the directories manually.

[2]`https://github.com/ASU-CompMethodsPhysics-PHY494/assignments-2019-`*`YourGitHubUsername`*

Homeworks must be legible and intelligible or may otherwise be returned ungraded with 0 points.

Proposal submission To submit your proposal (see below in Problem 10.2 for detailed instructions on format and content), commit and push a PDF file inside the `assignment_10/Submission` directory and name it `proposal.pdf`.

Collaboration is not allowed for this assignment. Each student must produce their own proposal.

Project pitch On **Thursday April 4, 2019**, you have your opportunity to *pitch your project to the class* and *attract a team of students* to work with you on the project. **Only projects that were pitched to the class can be chosen!** You will have 5 Minutes in total to present you project in an exciting and succinct manner.

If you want to **show slides** email a **PDF** to the instructor on the day before.[3] All slides will be queued up on the instructor's laptop to allow for presentations with minimal down time for changing laptops. However, slides are not required for the pitch — you can use the whiteboard or simply talk.

10.1. Square-root with Newton's method (15 points)

The square root function[4] $q(x)$ can be *defined* by the equation

$$q(x)^2 = x. \tag{1}$$

The goal is to develop and to implement an **efficient algorithm to compute square roots**.

[3] Also include the presentation in your `assignment_010/Submission` directory and submit it; it will not be graded but it is very convenient to have everything in one place and it will make it easy to get all presentations.

[4] The symbol $\sqrt{\cdot}$ is commonly used to denote the operation of q, so that $q(x) \equiv \sqrt{x}$. For the following it is more useful to think of the square root as a special function than just a calculation.

The defining equation Eq. 1 can be rearranged as

$$q(x)^2 - x = 0 \tag{2}$$
$$q^2 - x = 0 \tag{3}$$
$$f_x(q) = 0 \tag{4}$$

where $f_x(q) = q^2 - x$ is now considered a function of the *variable* q and a given *parameter* x. Finding the square root $q(x)$ amounts to finding the root of $f_x(q)$, i.e., find that value q that makes Eq. 3 true for a given x.

(a) Use the iterative Newton-Raphson algorithm from the class to implement a function `sqrt(x, tol=1e-6, Nmax=100)` in a module `functions.py` that returns the square root of x to a tolerance of `tol` and uses at a maximum `Nmax` iterations. If `Nmax` is exceeded print a warning message and return `None`.[5] Your code should guess a good starting value for the Newton-Raphson scheme, e.g,. $x/2$.

In the Newton-Raphson scheme you have to calculate the update Δq to q in order to obtain the new best guess for the root

$$q \leftarrow q + \Delta q \tag{5}$$

with

$$\Delta q = -\frac{f_x(q)}{f_x'(q)} \tag{6}$$

In your code you may either use a finite difference scheme to calculate $f_x'(q)$ *or* (more efficiently), use the explicit *analytical derivative* $f_x'(q) = \frac{df_x}{dq}$ directly.[6]

[5] In your code you may *not* use a library square root function such as `math.sqrt` or `numpy.sqrt` nor taking fractional powers such as `x**0.5` or `numpy.power(x, 0.5)`.

[6] Using the analytical derivatives makes for a handy algorithm to *manually* calculate square roots and indeed this is how Newton came up with the method. The algorithm for computing the square root was also already known to the ancient Babylonians.

Your code must produce correct results, as tested with `test_functions.py`.[7] You can run these tests yourself with

```
pytest -v test_functions.py
```

(in the same directory as your `functions.py`). **[10 points]**

(b) Show results for $x = 0, 0.456\times10^{-8}, 10^{-3}, 0.1, 0.64, 0.99, 1, 5, 9, 12.5, 10^3, 1.2345\times 10^8$ for a tolerance of 10^{-6}. Put the results in a text file `sqrt.txt`. The results should be arranged so that each line contains x and $q(x)$. **[5 points]**

(c) Bonus: Do one Newton-Raphson step analytically (using $f'_x(q)$) and come up with a handy way to calculate approximations to square roots. Apply your approximation formula to estimate the value of $\sqrt{2}$ and compare to the value obtained from `numpy.sqrt()` or your calculator. **[bonus +3*]**

Note: You don't have to submit any notebooks or written text for problems (a) and (b). It will be sufficient to submit

- `functions.py`
- `sqrt.txt`

If you are also solving the *Bonus problem (c)* then include a text document in PDF format to show your solution.

10.2. Proposal for the Final Project (20 points)

Write a short proposal for the *Final Project*. You can come up with your own idea or base your proposal on the list of suggestions in the Appendix A.[8]

[7] Some specific tests are allowed to fail and they are marked with `x` or `xfail` in the test output — this is ok.

[8] The Final Project, the proposal, and the suggested projects were discussed in class on 2019-03-28. The slides are available from Canvas and in the PDF `final_overview_2019.pdf` within the HW10 repository.

10.2.1. Formatting [5 points]

The proposal must adhere to the following formatting restrictions:

- 1 page maximum, 0.75 pages minimum length
- minimum font size 11pt
- minimum margins 1" on all sides
- include
  - **title**
  - **author** (your name)
  - sections with headings **Problem**, **Approach**, and **Objectives** (in this order); see below for what the content should be.

10.2.2. Content [15 points]

The proposal should concisely describe what project you want to undertake and suggest a roadmap for how you are going to do it. It should be written in *full English sentences*. Try to write clear and succinctly. If you can read it to someone not in the class and they get a general idea of what you want to do then you are on the right track.

Specifically, you must address the following points in the individual sections:

Problem Describe the problem to be solved: What is the background, what is the overarching question. What is the physics governing the problem, what are the important equations—show them, if possible, but at a minimum name them. You can also comment on why this is an interesting or difficult problem.

Clearly define the overall goal of what you want to find out.

This should be between 30-50% of your text.

Approach Describe *how* you are going to reach your goal, i.e., answer the overarching question. How are you going to solve the equations that you identified in the Problem section, what kind of calculations are needed, which algorithms are you going to use? Do you need input parameters? Where can you get them from (show URLs or sources if possible).

Be as concrete as possible: you want to convince your audience that it is feasible to solve this problem and you have an idea how to tackle it.

This should be between 30-50% of your text.

Objectives Use a numbered list to state 3–6 measurable non-trivial outcomes that you need to achieve in order to reach the overall goal. These are the milestones that you have to reach; they are possibly dependent on each other. For each objective it must be clear how to decide if you fulfilled it or not. Objectives are often formulated in terms of deliverables such as "Compile a list of material parameters for walls, windows and doors (heat conduction coefficients, typical thicknesses)."

Your grade will partially depend on achieving the objectives that you set here.[9]

This should take up your remaining space. The Objectives are very important. Make sure to formulate them clearly. They will be your own roadmap when you do the project.

A. Suggested projects

The following are suggestions that can be used to develop projects. The estimated difficulty level is only very approximate (1 being easiest and 4 hardest).

[9]The instructor will vet the objectives for any proposal that is being used as the basis for a Final Project.

A.1. Monte Carlo simulation of liquid argon

Argon can be simulated as a *Lennard-Jones* fluid. Using random numbers and the *Monte Carlo* (MC) approach we can simulate it at constant temperature.

- difficulty: 1–2
- implement basic MC for liquid Argon in the NVT ensemble (see *Computational Physics* and/or Frenkel and Smit, *Understanding Molecular Simulations*)
- use periodic boundary conditions with the minimum image convention
- analyze at different T and calculate the equation of state $P(T, \rho)$; look for a phase transition
- visualize
- extra work: implement NPT ensemble (with volume moves)

A.2. Monte Carlo simulation of the Ising magnet

The classical application for Monte Carlo methods and one of the best known hard but analytically solvable problems in statistical mechanics.

- difficulty: 2 (2D), 3 (with 3D)
- implement MC sampling for the Ising spin system in 1D, 2D, and possibly 3D
- see *Computational Physics* or *Computational Modelling and visualization of physical systems*
- compute magnetization $M(T)$ and look for a phase transition
- visualize

- compare to the exact 2D result (see eg Kerson Huang, *Statistical Mechanics*)

A.3. Classical chaotic scattering

Inspired by pinball flippers: what is needed for chaotic scattering, i.e., small initial differences in an incoming beam lead to large differences in the scattered beam.

- Difficulty: 1
- Problem 9.4 in *Computational Physics*
- particle interacting with “4-peak” potential
- integrate with RK4
- vary parameters and analyze cross section, assess sensitivity to initial conditions
- visualize trajectories

A.4. Quantum mechanical wave packet propagation

Simulate a realistic representation of particles interacting with various potentials and see first-hand how different quantum mechanical particles behave from classical ones.

- difficulty: 3
- solve the time-dependent Schrödinger equation for a wave packet interacting with various potentials (e.g. using the explicit Visscher algorithm described in *Computational Physics* 22.2.1 or the Maestri/Askar & Cakmak algorithm (*Computational Physics* 22.3))
- 1d

  - step barrier (vary height compared to wave packet energy and observe tunneling)
  - harmonic well
  - square well
- 2d: observe interference *for particles*:
  - single slit
  - double slit
- calculate transmission coefficient, wave velocity, probability density on a screen behind slits
- visualize

A.5. Fluid dynamics: 2D Navier-Stokes equations

The Navier-Stokes equations govern all of atmospheric and ocean physics, are needed to design airplanes and ships, and are fiendishly non-linear. But you can solve them with a computer:

- difficulty: 2–3
- Use *Computational Physics* Chapter 25 with code as starting point
- Solve the Navier-Stokes equations for 2d flow (using finite difference with successive over-relaxation)
- investigate the velocity (flow) field and vorticity around various submerged objects:
  - beam
  - sphere/cylinder
  - drop shape
- visualize
- vary Reynolds number: effect on vorticity?

A.6. Dynamics of the solar system

Simulate the whole solar system including a number of comets.

- difficulty: 2
- simulate the solar system (classical gravitational interactions) with velocity Verlet
- include sun and planets and comets such as Halley's comet or the Shoemaker-Levy comet and let it crash into Jupiter
- find appropriate parameters (masses, positions, periods)[10]
- investigate
  - stability of the solar system
  - changes in orbits due to interactions (probably to weak to easily notice?)
  - Do comets crash into planets?
  - extra: course of small space craft (e.g. a gravity assist swing-by maneuver)
- extra: Make Jupiter like the sun and see how it changes the solar system.
- extra: build the Tatooine system: binary star (two suns) and an earth-like planet ("Tatooine"). Does the planet have stable orbits? What would the days on the planet look like? Is there an orbit that allows life?
- visualize with vpython[11]

[10]See NASA HORIZONS to get data: http://ssd.jpl.nasa.gov/horizons.cgi
[11]For a great example see the animated orrery http://mgvez.github.io/jsorrery/

A.7. Wavelet analysis of Stock markets or Oil prices

Econophysics is a major employer of physicists... apply physical ideas to markets. If you can predict where the prices go you can make a lot of money...

See for instance the paper "Wavelet-based prediction of oil prices", S. Yousefi, I. Weinreich, D. Reinarz, *Chaos, Solitons and Fractals* **25** (2005), 265–275, doi: 10.1016/j.chaos.2004.11.015

- Difficulty: 4
- Implement wavelet analysis (Daubechies) for stock prices and commodity prices time series. (see *Computational Physics*)
- Test how well you can forecast known data: correlate forecast vs actual data (which was not used for the wavelet analysis). How does the correlation coefficient of the forecast with the real timeseries vary as a function of the forecasting time interval?

Disclaimer: If you make money from this project, it's all yours.

A.8. Agent-based modelling of self-driving cars

The future belongs to self-driving cars, which will communicate with each other. Once all cars are self-driving they might behave like a swarm of birds or bees or a school of fish: simple and local rules (such as keep a minimum distance, move if your neighbor moves, ...) can lead to seemingly complex behavior in aggregate. Are self-driving cars going to improve traffic (e.g., increase traffic flow throughput)? How likely are accidents if small errors occur, i.e., how robust is the system?

I heard something like the following in a TED talk:

> "Once cars can talk to each other, we will not need any traffic lights or even restrictions on which side of the road they can drive, it will look very organic, e.g. like water flowing."

Is this true?

- difficulty: 2 (?)
- Use *agent-based modelling* to simulate large numbers of cars on a small street network. (Basically, many little car objects that all follow their own rules in response to everything else in the environment.)
- Analyse traffic flow as function of car density (and rules).
- Extension: introduce random errors or human drivers and study robustness.

A.9. Analysis of natural motion

When you look at a film of a human walking, a cheeta or a horse racing, a bird flying, a fish swimming, or a snake slithering you get 24 pictures per second that are all different and seem to show a very complicated process. However, using machine learning techniques such as *singular value decomposition* (SVD — see lecture 14!) these movements can often be decomposed into a small number of components and one gets a quantitative handle on the motion. (Example: Girdhar K, Gruebele M, Chemla YR (2015) The Behavioral Space of Zebrafish Locomotion and Its Neural Network Analog. *PLoS ONE* 10(7): e0128668. doi:10.1371/journal.pone.0128668)

- difficulty: 3–4 (?)
- record or obtain video of animal/human motion and process it with Python
- use SVD to decompose motions and assess the complexity
- What are the most complex motions? Is human walking more complex than a fish swimming or an eagle flying? Do all birds fly the same way?

- Extra: Perform "gait analysis/recognition" (also see *Misson Impossible 5: Rogue Nation*[12]): can the way you walk act as a unique finger print?

[12]See https://www.youtube.com/watch?v=0iZ-nQ4yFn4 starting at 0:51.