6 — PHY 494: Homework assignment (20 points total)

Due Sunday, Feb 24, 2019, 11:59pm.

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_06/Submission`, `assignment_06/Grade`, and `assignment_06/Work`.

2. You can try out code in the `assignment_06/Work` directory but you don't have to use it if you don't want to. Your grade with comments will appear in `assignment_06/Grade`.

3. Create your solution in `assignment_06/Submission`. Use Git to `git add` files and `git commit` changes.

   You can create a PDF, a text file or Jupyter notebook inside the `assignment_06/Submission` directory as well as Python code (if required). **Name your files `hw06.pdf` or `hw06.txt` or `hw06.ipynb`**, depending on how you format your work. Files with code (if requested) 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.

[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`*

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

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

This assignment contains **bonus problems**. A bonus problem is optional. If you do it you get additional points that count towards this homework's total, although you can't get more than the maximum number of points. If you don't do it you can still get full points. Bonus problems and bonus points are indicated with an asterisk "*".

For problem 6.2: If you implement the function as specified you can run the tests in the file `Submission/test_hw006.py` with `pytest`

```
cd Submission
pytest test_hw06.py
```

and all tests should pass. If you have errors, have a look at the output and try to figure out what is still not working. Having the tests pass is not a guarantee that you will get full points (but it is general a very good sign!).

6.1 BONUS: Discussion of the errors of finite difference operators (7* bonus points)

In lesson 09 Differentiation you plotted the absolute error $|E_h(t)| = |D_h \cos t - \cos t|$ for three different algorithms for the finite difference operator D_h with step size h (forward difference, central difference, and extended difference) and for three different values $t = 0.1, 1, 100$.

Complete all calculations and compare your plots to the ones shown at the end of the notebook 09-differentiation.ipynb. Discuss the following questions (write them in a simple text file `problem1.txt` or you can also submit a notebook `problem1.ipynb` but make clear where you answer the questions below):

(a) Which algorithm produces the most accurate[3] result? Give order-of-magnitude estimates for each one, based on your data. **[bonus +2*]**

(b) Describe the general shape and features of your graphs. Explain why you see increases and decreases in accuracy with varying h. **[bonus +2*]**

(c) What is the best value of h in each case? How does the best value of h change with the algorithm? Explain the observed behavior in the light of the answer to your answer (b). **[bonus +3*]**

6.2 Exponential function (20 points)

The exponential function has the series expansion

$$\exp x = \sum_{n=0}^{+\infty} \frac{x^n}{n!} \tag{1}$$

An algorithm to compute $\exp x$ makes use of the iterative solution[4]

$$a_n = \frac{x^n}{n!} \tag{2}$$

$$a_{n+1} = a_n q_{n+1} = \frac{x^{n+1}}{(n+1)!} = \frac{x^n}{n!}\frac{x}{n+1} \tag{3}$$

$$q_n = \frac{x}{n+1} \tag{4}$$

(a) Create a function `exp_series(x, eps=1e-15)` in a file `problem2.py` that computes the $\exp(x)$ function based on the series expansion

[3] The *accuracy* is measured by the deviation from the exact result, i.e. the lower $|E|$ the more accurate. *Precision* measures how well a number is defined and is essentially the machine precision in our case.

[4] Similar to the iterative solution for the $\sin x$ function that was discussed in Lecture 07.

Eq. 1 and the iterative solution Eq. 2–4. The function should only return the value of $\exp(x)$.

The function should take an argument `x` and optional convergence criterion `eps` with default `1e-15`.

The iteration should stop when the **convergence criterion**

$$\left| \frac{a_N}{\sum_{n=0}^{N} a_n} \right| \leqslant \epsilon \tag{5}$$

is fulfilled. **[12 points]**

(b) Show results for $\epsilon = 10^{-15}$ and $x = -9.2103437, 0, 1, 100$ **[4 points]**

(c) Show results for $\epsilon = 10^{-4}$ and $x = -9.2103437, 0, 1, 100$ **[4 points]**

6.3 BONUS CHALLENGE: Stable sum (+10* points)

This problem is a bonus problem that challenges you to do your own research and come up with a working solution.

The normal Python `sum()` or `numpy.sum()` functions can easily loose precision. As an example, try calculating the sum over the sequence

$$S_M = \{\underbrace{1, 10^{100}, 1, -10^{100}}_{\text{repeated } M \text{ times}}, \ldots, 1, 10^{100}, 1, -10^{100}\}$$

which is

$$\sum_{x \in S_M} x = 2M. \tag{6}$$

In Python you can generate S_M with the "list multiplication" operation

```
M = 10000
SM = [1, 1e100, 1, -1e100] * M
```

(which generates a new list that consists of the concatenation of `M` copies of the original list).

(a) Convince yourself that `sum()` and `numpy.sum()` give the wrong answer for Eq. 6 and $M = 10000$. Show the results. [**bonus +1***]

(b) Show that `math.fsum()` gives the correct answer. [**bonus +1***]

(c) Implement an algorithm in a function `stable_sum()` in a file `problem3.py` that takes a sequence of numbers as input and calculates the sum Eq. 6 correctly. You cannot use `math.fsum()`. If you implement an algorithm from the literature, cite your sources. Show the output of your algorithm for the sum for $M = 10,000$ and $M = 10^6$. [**bonus +8***]