3 — PHY 494: Homework assignment (28 points total)

Due Thursday, Feb 9, 2017, 1:30pm.

Submission is now to your **private GitHub repository**. Follow the link provided to you by the instructor in order for the repository to be set up: It will have the name *ASU-CompMethodsPhysics-PHY494/assignments-2017*-YourGitHubUsername and will only be visible to you and the instructor/TA. Follow the instructions below to submit this (and all future) homework.

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

1. `git clone` your assignment repository (change *YourGitHubUsername* to your GitHub username)

   ```
   repo="assignments-2017-YourGitHubUsername.git"
   git clone https://github.com/ASU-CompMethodsPhysics-PHY494/${repo}
   ```

2. run the script `./scripts/update.sh` (replace *YourGitHubUsername* with your GitHub username):

   ```
   cd ${repo}
   bash ./scripts/update.sh
   ```

   It should create three subdirectories[1] `assignment_03/Submission`, `assignment_03/Grade`, and `assignment_03/Work`.

3. You can try out code in the `assignment_03/Work` directory but you don't have to use it if you don't want to. Your grade with comments will appear in `assignment_03/Grade`.

4. Create your solution in `assignment_03/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_03/Submission` directory as well as Python code (if required). **Name your files `hw03.pdf` or `hw03.txt` or `hw03.ipynb`**, depending on how you format your work. Files with code (if requested) should be named exactly as required in the assignment.

5. 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-2017-`*`YourGitHubUsername`*

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 "*".

3.1 Representation of numbers in the computer (6 points)

(a) Convert the decimal number -2.65625_{10} to binary. What is the binary representation and how many bits do you need? [**2 points**]

(b) Convert the decimal number $+0.8_{10}$ to binary. What is the binary representation and how many bits do you need? [**2 points**]

(c) What is the smallest and largest *unsigned integer* that can be represented with 8 bits (= 1 byte)? [**2 points**]

(d) Bonus: In the following, show how you arrive at your answer and give answers in standard scientific notation (i.e., in powers of ten):

   (i) Using the information in the notebook 07-numbers.ipynb and the representation $(-1)^s \times 2^{e-1023} \times 1.f$, derive the largest value that can be represented with an IEEE 754 *double* floating point number. Note that in IEEE 754, the highest possible value $e_{\max}$ for the exponent e is reserved and *not* used; instead the highest value is one less, $e_{\max} - 1$. [**bonus +3***]

   (ii) For the smallest (in absolute value), non-zero IEEE 754 *double* numbers, $e = 0$ and the value is defined as $(-1)^s \times 2^{e-1022} \times 0.f$. What is the smallest non-zero, absolute value that an IEEE *double* can represent?[3] [**bonus +3***]

(e) Bonus: Determine experimentally the machine precision ϵ_m for numpy 128-bit floats (`dtype=np.float128`). Show your code[4] and result. [**bonus +3***]

3.2 Errors in numerical summation (22 points)

In class and in the notebook 07-solution-sine-series.ipynb an algorithm to compute $\sin x$ was derived, based on the infinite series

$$\sin x = \lim_{N\to+\infty} \sum_{n=1}^{N} \frac{(-1)^{n-1} x^{2n-1}}{(2n-1)!}. \qquad (1)$$

[3] The answer is not 0 (looking for non-zero) and not "minus the answer from the previous question" (looking for the absolute value).

[4] Note that this is tricky to get right: all numbers must be of the same type because otherwise numpy coerces types to (typically) the most inclusive one.

In this algorithm, the nth term a_n was computed from the $(n-1)$th term a_{n-1}. As convergence criterion we demanded that

$$\left|\frac{a_n}{\sum_{k=1}^{n} a_k}\right| < \epsilon_m. \tag{2}$$

You have a working implementation in the function `sin_series()` in the notebook.[5]

(a) Plot (1) your result and the (for our purposes) exact solution from `np.sin()` as a function of x, (2) the maximum `n` used in the calculation, (3) the relative error

$$\delta = \left|\frac{\text{sin\_series}(x) - \sin x}{\sin x}\right|, \tag{3}$$

and (4) the absolute error

$$\Delta = \left|\text{sin\_series}(x) - \sin x\right|. \tag{4}$$

(You can use the existing code as a starting point and add a new calculation for Δ and a new plot.)

  (i) Plot in the range $-2\pi \leqslant x \leqslant 2\pi$ (step size 0.1) and **[2 points]**

  (ii) in the range $-50\pi \leqslant x \leqslant 50\pi$. **[2 points]**

  (iii) Make a table with headings x, $\max n$, sin_series, δ, Δ and show results for five values of x that, in your opinion, best illustrate and quantify the behavior seen in the plots. **[3 points]**

(b) Show that for sufficiently small values of x, your algorithm converges (the changes are smaller than the tolerance level) and it converges to the correct answer. (Use your results from (a) and possibly calculate additional specific values of x and add the results to the table.) **[3 points]**

(c) Compare the number of decimal places of precision obtained (δ) with that expected from Eq. 2. **[2 points]**

(d) Show that there is a range of larger values for which the algorithm[6] converges but that it converges to the wrong answer. Calculate any results that you need for your answer and add them to the table. **[2 points]**

(e) Now make use of the identity $\sin x = \sin(x + 2n\pi)$ (where n is any integer) to compute $\sin x$ for large values of x where the algorithm would otherwise diverge. Show your code. Show the plot for $-50\pi \leqslant x \leqslant 50\pi$ and also show results for five values of x that illustrate the improvement in behavior. **[4 points]**

[5] Try the example values and plotting routines in the notebook. Use the code as a pattern for your own work.

[6] Consider the algorithm as implemented, namely without using the identity $\sin x = \sin(x + 2n\pi)$.

(f) Implement the "bad algorithm" for calculating $\sin x$ whereby you calculate each term a_n explicitly with factorials (use `math.factorial()` in the `math` library) and powers in x. Show your code. Compare the "good" and the "bad" algorithms by showing plots across a range of values where both algorithms give results and by showing some results explicitly in a table as above. Briefly discuss your results. [**6 points**]