P395: Guide 2:

Start up a new notebook and title it 'Guide2_your_name'. We won't be using interactive plots this week, so you can use

```
import matplotlib.pyplot as plt

import numpy as np
```

Remember to put in helpful headings, etc., to keep your notebook organized.

Task 1: Numerical error

Types of error

This week we begin to use numerical approaches to solve problems for which finding an analytic solution is difficult. Whenever solving a problem on a computer, one is confronted with the challenge of not being able to compute things exactly and thus introducing error into the calculation. There are two main types of error that one encounters when doing numerical calculations: truncation error and rounding (round-off) error.

1. Analysis: Look up truncation error. Summarize what it is in a couple sentences in a text cell. Give a mathematical example that would lead to truncation error on the calculated result.
2. Analysis: Look up rounding error. Summarize what it is in a couple sentences in a text cell.
3. Estimate what the machine epsilon is by calculating '`7./3 - 4./3 - 1`' in your notebook.

NOTE: The error associated with the above calculation comes from doing floating-point calculations and how real numbers get represented in the computer. (For those that are interested you can google to find out why this works, but something like `4./3 - 1./3 - 1` does not (it computes to 0). Specifically, it has to do with how floating-point numbers are represented in binary.)

How to evaluate error in a calculation:

There are various ways to define error. For real-valued variables one can define the absolute error as

$$\epsilon_{abs} = |y_0 - y| = \sqrt{(y_0 - y)^2}$$

where y_0 is the true value and y is the calculated value. Another measure is relative error,

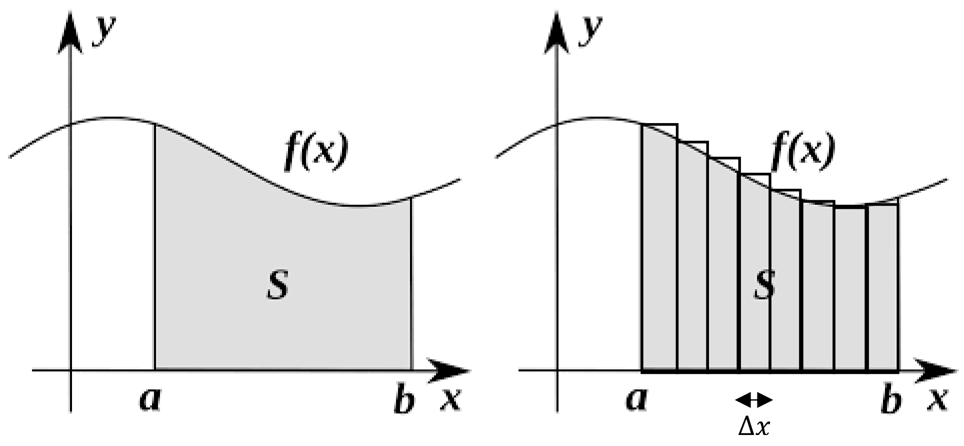
$$\epsilon_{rel} = \left|\frac{y_0 - y}{y_0}\right|$$

(In calculations where one is checking the error often, one will typically evaluate the squared error to avoid calculating square roots which slow things down.)

Numerical Integration:

Task 2: Riemann Sum

Many physics problems require us to find the integral of some function over a given interval. In its simplest terms, calculating an integral is the same as finding the area under the curve. One can imagine using shapes of known area, fitting them under the curve and then adding up their areas to get an estimate of the integral. All numerical integration methods carry out such a procedure, by using either simple or sophisticated ways to estimate areas and then adding them up. In the exercises that follow, you will hopefully appreciate the nature of these approximations and when to use which method. Just note that all these methods are approximations to the true integral and so can fail if not used correctly.



Consider the above function over the interval $a \leq x \leq b$. A simple way to find the area is just to divide into *N* rectangles of uniform width $\Delta x = h = (b - a)/N$ and add up their areas. Performing such a construction, the integral is approximated by the Riemann sum

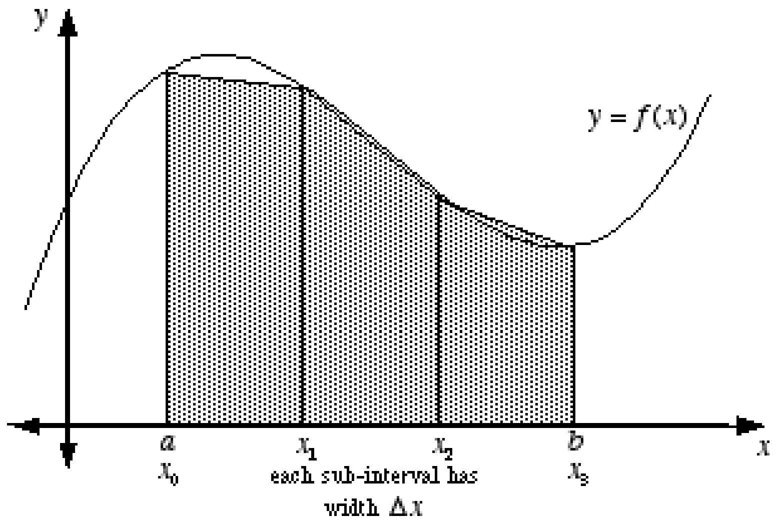
$$\int_a^b f(x)\mathrm{d}x \approx \sum_{i=0}^{N-1} f(x_i)\, h$$

1. Write up a function that carries out the above approximation to the integral of a function over an interval $a \leq x \leq b$ using *N* rectangles and where you use the function value at the *left side* of each rectangle as the height (as shown in the figure). This is known as the *left Riemann sum*. Your function should take as arguments an array of values of the function at the given points and the slice width *h*.
2. Evaluate the integral of $\sin\, x$ over $0 \leq x \leq \pi/2$ using different numbers *N* of slices. Plot your absolute error (you'll have to figure out the exact value by doing the integral analytically) as a function of bin width *h*. Use a logarithmic scale of *N*, say *N* from 10^1 to 10^6.
3. Analysis: If the error goes as $\text{error} = Ah^{\alpha}$, then how does the logarithm of the error depend on the logarithm of the bin width *h* (i.e., what type of function is log(error) ?) ? Write your equation in a text cell.

4. Determine the power dependence of your error on the bin width *h* by fitting a polynomial of the appropriate degree to log(absolute error) vs log(*h*). To do the fit, look up the function `np.polyfit`.
5. Analysis: What do your fit parameters tell you about the power dependence of the numerical error on *h* for the Riemann sum method?
6. Analysis: To get an error of 1e-8, estimate how many slices you would need to use with the above method.

Task 3: Trapezoid Rule

There are many ways to make improvements to the above method. They all involve making better estimates for the areas. The first method is to use a linear interpolation between function points, making the area a trapezoid rather than a rectangle. Look at the image below which shows how the trapezoid rule is constructed,



1. Analysis: Using the plot above, derive the resulting approximation to the integral by adding up the areas of the 3 slices. HINT: each slice is composed of a rectangle plus a triangular portion on top. In a text cell, write up your equation for the approximation to the integral.

Using *N* slices, the construction above uses the function evaluated at *N*+1 points, and leads to the trapezoid rule,

$$\int_a^b f(x)\mathrm{d}x \approx \left[\frac{f(x_0)+f(x_N)}{2} + \sum_{i=1}^{N-1} f(x_i)\right] h$$

(NOTE: we have *N*+1 points, corresponding to the *N* slices or areas. Be careful when coding your own numerical integration schemes to get your *N* correct.)

2. Import the `scipy.integrate` module (e.g., `import scipy.integrate as integrate`) as it contains a number of useful integral approximation formulas, including the trapezoid rule.
3. Look up the help on how to use the trapezoid rule in this module.

4. Now do the same calculation as in Task 1, but using the `scipy.integrate` trapezoid rule to calculate the absolute error at different values of *h* (i.e., *N*) for sin(x) over $0 \leq x \leq \pi/2$.
5. <u>Analysis</u>: Is the error lower for a given *h* than the Riemann sum method above?
6. Determine how the error depends on *h* for the trapezoid rule by, again, fitting a line to the logarithm of the errors vs. the logarithm of *h*.
7. <u>Analysis</u>: Estimate what *h* and corresponding *N* you would need to use to get a numerical error of 1e-8 now using the trapezoid rule?

Task 4: Simpson's Method

The next approximation to improve the area estimates is to use 3 points instead of 2 and approximate the function using a parabola between the 3 points rather than a line between 2 points as in the trapezoid rule. This method is due to Simpson and is so named. The algebra to derive Simpson's rule is a bit involved, so the result is quoted for when **<u>the number of slices</u>, *N=2n*, is an even number, so the total number of points is odd** (the notation here is $f(x_i) = f_i$),

$$\int_a^b f(x)\mathrm{d}x \approx \frac{h}{3}\left[f_0 + 4(f_1 + f_3 + \cdots f_{2n-1}) + 2(f_2 + f_4 + \cdots f_{2n-2}) + f_{2n}\right]$$

1. Look up the help on how to how to use Simpson's method in `scipy.integrate`.
2. Carry out exactly the same analysis as in Task 2, plotting the absolute error at different values of *h* (i.e., *N*). HINT: you may need to use a smaller range of *N* here.
3. Determine the order of error dependence on *h* for Simpson's method by fitting the errors, as above.
4. What *h* (and corresponding *N*) do you need now to get an error of 1e-8.
5. Do a timing of the three approaches (simple method, trapezoid rule, and Simpson's method) to get the same absolute error of 1e-8. Which method ends up being the fastest?

Task 5: Gaussian Quadrature

In optics, light diffracting around opaque objects leads to an integral known as the Fresnel integral,

$$C(x) = \int_0^x \cos\left(\frac{\pi}{2}x^2\right)\mathrm{d}x$$

1. Plot $\cos(\pi x^2/2)$ over the range $0 \leq x \leq 7.1$.
2. <u>Analysis</u>: Do you think this integral could give challenges to the above approaches that are based on using uniformly spaced points? Write your answer in a text cell and give a short reason as to what the challenges might be.
3. Learn how to import the Fresnel integral function from `scipy.special`. Use it to evaluate the above function at *x*=7.1. (Note: scipy's function returns values for both the sin and cos Fresnel integrals).
4. Use Simpson's method, and determine what *h* (and corresponding *N*) you need to use so that the absolute numerical error is less than 1e-8 using the calculated value in (3) as the true value.

This shows that methods based on uniform points can be problematic.

A method known as *Gaussian quadrature* (see Wikipedia or Numerical Recipes 4.6 for more info) uses non-uniformly spaced points to evaluate the integral. In short, if doing quadrature of order *N*, it finds *N* points in the domain such that if the function were a polynomial of degree less than or equal to 2*N*-1, then the resulting approximate formula would be exact. (The downside of quadrature is that you need to have a mathematical expression for the function you are trying to integrate. In many situations we won't know the function directly and will just have it evaluated at a set of points, in which case we can use the other methods.)

5. Look up how to use `scipy.integrate.fixed_quad` and determine what *N* is needed to get an error < 1e-8 for the Fresnel cosine integral above.

Lastly, there is a general-purpose function `scipy.integrate.quad` which calls Fortran's QUADPACK library to carry out the integrals. It can even do integrals ranging from $x = \pm\infty$. This is the function I would use to evaluate most integrals numerically.

6. Look up the help on `scipy.integrate.quad` and use it to integrate the function $\exp(-x^2/2)$ from $x = -\infty$ to $x = \infty$. The exact answer is $\sqrt{2\pi} = 2.506628274631$.

Task 6: Monte Carlo integration

Background:

One last integration method that is distinct in its approach from the above methods is known as Monte Carlo integration. We will encounter it again when looking at problems in statistical physics. Imagine we have some function $f(x)$ that we want to integrate over the interval $a \leq x \leq b$. Using the definition of the average $< f >$ of the function over the region,

$$\int_a^b f(x)\mathrm{d}x = (b-a)\langle f \rangle$$

Using this, we can compute the integral if we can reliably estimate the average value of the function over the region. We could simply do this by sampling *N* points $\{x_i\}$ uniformly over the region and computing the average

$$< f >= \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

The advantage of Monte Carlo sampling is that it can reliably estimate the average with only a few samples. This is particularly useful for integrals in higher dimensions where it is typically the best approach. However, it has drawbacks. If the function is pathological, many samples may be required in order to get the desired tolerance. Another drawback is that it can be horribly inefficient. For example, if we have a function that (unknowingly) happens to be zero over a large part of the domain, we end up wasting time sampling from these regions. This leads to techniques that compensate for this known as *importance sampling*. We will see these techniques later in the course.

1. Use Monte Carlo integration to evaluate the Fresnel integral above using different numbers *N* of samples. Use a logarithmically spaced set of *N* from 10^1 to 10^6.
2. Plot the absolute error of your integral vs. *N*. Set your axis scales to be logarithmic.

3. Analysis: Is the error trending down with *N*? Can you see any issues with using Monte Carlo integration for doing 1D integrals? Put your answer in a text cell.

Monte Carlo for multidimensional integrals:

Now you will do a calculation where you will see the usefulness of Monte Carlo integration. Consider calculating a multidimensional integral, such as the volume of a 5D sphere.

4. Use Monte Carlo integration to find the volume of a 5-dimensional hypersphere with radius *R*=1, using different numbers *N* of samples. (HINT: You will need to randomly sample points uniformly from a 5-dimensional cubic volume. The function you are calculating the average of has a value of 1 everywhere inside the 5D sphere and a value of 0 everywhere outside the sphere.)
5. Look up the exact formula for the volume of a 5D sphere and calculate the true value using *R*=1.
6. Plot the absolute error again versus the sample size. Roughly, how many points do you need to get an absolute error of 1e-3?

NOTE: If you were to evaluate the volume integral using the grid-based methods above (such as trapezoid and Simpson's rule), you would have to sum over a 5-dimensional grid. The number of function calls would scale as $(N_{\text{grid}})^d$, where N_{grid} is the number of points along each axis and *d* is the dimension. This makes the grid-based approaches computationally very expensive and favours doing Monte Carlo sampling, where many fewer points can be used to achieve reasonable accuracies.

Task 7 – Wrap up

Please provide any feedback you might have on the Activity guide. Using text cells, please provide answers to the following questions:

1. Were there any questions that were confusing or worded in a way that made them difficult to understand? If so, please reference the Task and question number, so that I can improve it.
2. Please provide any other feedback that you think might have helped to improve your learning experience with this Activity Guide.
3. Download your notebook to the local computer and email it to yourself for safe keeping (i.e., make a backup). Don't assume that your files will be there the next time you log in to syzygy (but let's hope so).