Assignment 7: Integration

Revision 1

1 Problem statement

A common operation that we are faced with in computational mathematics is the calculation of the definite integral for a function $f(x)$.

$$\int_a^b f(x)\ dx$$

For some functions it is easy to calculate the integral symbolically, but sometimes we are given functions where symbolic integration is not possible and we must resort to numerical methods. In this assignment we will code up a simple Monte Carlo integrator and explore its performance on functions that we can calculate an exact integral for.[1]

1.1 Monte Carlo integration

Given a function $f(x)$ and a set of n random points in the interval $[a, b]$, $x_1, \cdots, x_n$, we can estimate the integral as:

$$\int_a^b f(x)\ dx \approx \frac{b-a}{n} \sum_{i=1}^{n} f(x_i) \tag{1}$$

[1] In practice, Monte Carlo integration is applied to functions in high dimensions where other techniques like applying the trapezoidal rule don't scale well.

2 Tasks

We will consider the function:

$$f(x) = x^3 \sin(4x) \tag{2}$$

over the interval $[1.0, 7.0]$.

2.1 Compute the symbolic definite integral

Using SymPy, calculate the symbolic integral of $f(x)$ over the given interval $[a, b]$.

2.2 Implement a monte carlo integrator

Implement a function `mcintegrate` that takes four arguments: the lower and upper bounds of integration, the number of samples to take, and the function to integrate. This function should implement Equation (1) above. Tips:

- Use the `numpy.random.rand()` function to sample a random value from the interval $[0, 1]$, and transform that value to the interval $[a, b]$ to use to evaluate the function.
- Your function will contain a loop that iterates some number of times. Make sure to get a new random value to evaluate the function at on each iteration!
- Start your function with the line `numpy.random.seed(1)`. This will ensure that every time you run it, you get the same pseudo-random number sequence. This is very helpful for debugging and reproducibility.

2.3 Study the error as a function of sample count

Write a loop to run your `mcintegrate` function on $f(x)$ defined in Equation (2) for an increasing number of sample counts starting at 1000 and going to 100000 samples at increments of 1000 samples. Provide a plot showing the absolute value of the difference between the exact integral value and the value calculated by your Monte Carlo method. Make sure the axes are appropriately labeled and the x-axis tick marks correspond to the count values. **Note:** given that your function is using random samples, do not be surprised if your error curve doesn't look smooth. This technique assumes large sample counts are used in practice.

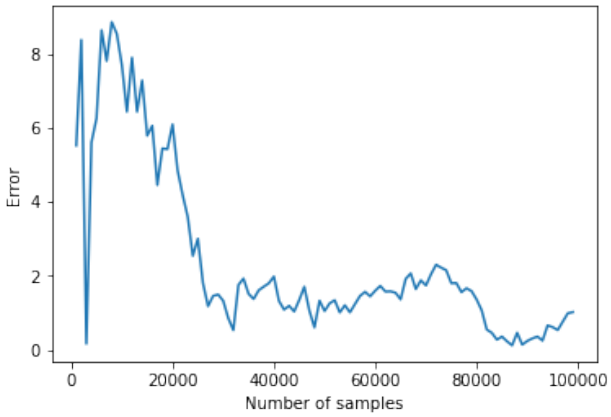

Figure 1: Example error plot.

3 Turn in

Turn in a Jupyter notebook containing your `mcintegrate` function, an error plot like the one shown above, and commentary in Markdown explaining what you did and what you found. Feel free to include any additional experimentation, either trying out different functions than the one provided above, varying the seed used to initialize the random number generator, and so on.