Monte Carlo

In this file we have provided a brief instruction for the program, its compilation, features, limitations and the probable to-dos to eliminate them or extend the program.

Introduction to Monte Carlo algorithm

Monte Carlo algorithm for integration is simply sampling over the domain of integration to estimate the mean value of the function. There are two main procedures to this problem: one is to uniformly choose samples in domain (uniform sampling) and the other is to choose the samples with respect to a weight function that have similar distribution to original function, so at the end, those parts have more contribution in the integral are considered more (importance sampling).

Compiling test suite

Inside *PCSC2017\_Group2/tests/* there are four CPP files, each evaluating one of the features of Monte Carlo integration.

* Test: Evaluates three integral numerically with both methods.
* StatisticalMomenst: Evaluates statistical moments of Gaussian distribution up to power 10 using uniform sampling. The answers will be written in a file. **Figure 1** shows the comparison between true and computed values.
* CentralLimitTheory: Integrates the same function many time to find the distribution of answers. Figure 2 Shows the distribution of N=? runs and the fitted normal distribution.
* Errors: Evaluates the true and estimated error for a specific function.

Program flow

For running the program, one should first specify the followings outside of the main block:

1. function that should be integrated. The function must get a double and returns a double
2. the non-negative weight function (if Metropolis method is chosen). Next step is to specify the method of integration by creating objects of the following classes inside the main block:
3. MonteCarlo\_UniformSampling
4. MonteCarlo\_MetropolisAlgorithm

After that, user have to link the function and its weight to those defined outside the main by using the following methods:

1. SetFunction(\*double): The argument is the function defined outside of main
2. SetWeight (\*double,bool) : The first argument is the function defined outside of main. The second argument is a flag indicating that if weight function is normalized (true) or not (false). If user is not sure about normalization, he/she can omit the second argument. By default, it is set to false to normalize the weight. Yet it is better to feed this method with normalized weight to avoid normalization error.

Next step is specifying the upper and lower limit of integration. the moment (if it is desired), and the number of sampling by using the following methods:

1. SetLowerLimit (double): Specifies lower limit.
2. SetUpperLimit (double): Specifies upper limit.
3. SetMoment (int): Specifies the moment number.
4. SetSamplingNumber (int): Specifies number of samples.

Finally, the method below, will return an array which contains the value of the integral in the first element and the evaluated error in the second one:

1. Integrator(): Computes integral with respective method

List of features

1. Integrating R to R functions inside a bounded domain using Uniform Sampling and Importance Sampling (using Metropolis procedure)
2. Computing different statistical moments of a function (integer or non-integer)
3. Normalizing weight function if needed only by taking a flag
4. Estimating error of integration
5. Being able to increase the accuracy by taking more samples as user's wish Integration in unbounded domain (should be used with caution, not recommended for all functions)

List of tests

We have tested program under the following benchmarks:

1. Computing Pi=3.14159...
2. Computing normal distribution integral
3. Computing gamma-function for z=4
4. Evaluating error and comparing them with true error for all integrals above
5. Computing even-moments of normal distribution up to 10

To-dos and prospective

The program can be extended to cover the following objectives:

1. Integrals from R^n to R
2. Integral in unbounded domain. {We've already provided such feature. Yet, we decided to comment it for the following reason. Using Monte Carlo algorithm on unbounded domain, is only possible with importance sampling (Metropolis) method. For using Metropolis, one should provide a suitable weight function considering the mathematical behavior of the original function. Also it is crucial to choose a good starting point to be able to span most of the subsets that contribute the most in the value of integral. These considerations cannot be implemented without pre-knowledge about shape of the function. In most cases, though, considering sufficient large upper and lower limit will give a reasonable answer.}
3. Active (self-corrector) step-size for Metropolis method