Year 3 Particle Physics Computing
Integration Methods Project Description

Resource 1 *Introduction to Monte Carlo Techniques* by Torbjörn Sjöstrand

Resource 2 *Introduction to Monte Carlo methods* by Stefan Weinzierl

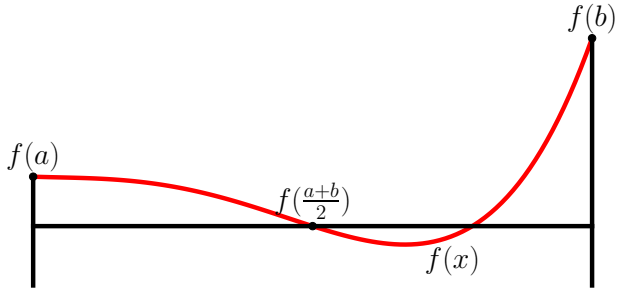
Resource 3 *PYTHIA 6.4 Physics and Manual* by Torbjörn Sjöstrand, *et al*; section 4

Resource 4 *Numerical Analysis* by Burden and Faires; chapter 4

Resource 5 *VEGAS: An Adaptive Multi-dimensional Integration Program* by Lepage

Typical events produced within the Large Hadron Collider (LHC) from colliding protons have $\mathcal{O}(100)$ or more particles produced. In Problem Set 2 we calculated a cross-section for a two-to-two process which required integrating over two variables, θ and ϕ. A two-to-n process requires integrating over $3n-4$ variables, so a typical LHC event would require integrating over $\mathcal{O}(300)$ variables. This is numerically challenging, at best, and with current technology is just simply not possible. To calcululate LHC events, we can instead factorise the problem into more manageable parts using probabilistic methods. Even still, calculating a perturbative cross-section for a 4-body final state requires integrating over 8 variables which is a challenging numerical integration. The bottom line is that performing high dimension integrals quickly and efficiently is a core problem in particle physic, and is very numerically challenging. In this project you will explore numerical integration techniques.

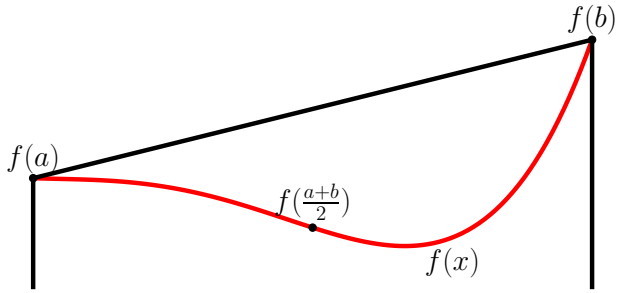
Consider a function $f(x)$ and its integral over the range a to b. One of the simplest non-Monte Carlo methods for performing this integral is to evaluate the function at its midpoint between a and b, and then multiply this by $b-a$.



$$\int_a^b \mathrm{d}x\, f(x) \approx (b-a) f\left(\tfrac{a+b}{2}\right) \tag{1}$$

This method is called the midpoint rule. Note that the midpoint rule requires just a single function evaluation. If the function is computationally expensive to compute, it is important to reduce the number of function evaluations. Of course, the midpoint might not provide a good estimate of the function, so instead we can divide the integral into multiple peices and for each division evluate the midpoint rule. This is the composite midpoint rule.

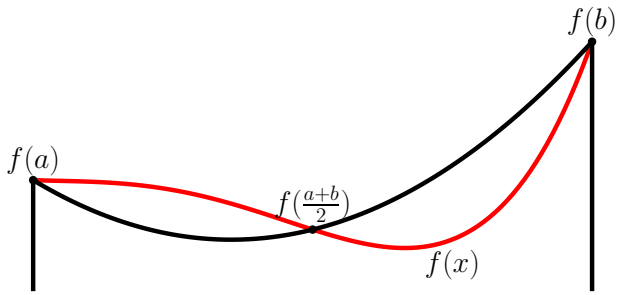
The midpoint rule approximates the function as a zero order polynomial, *e.g.* just a horizontal line, for which the integral is known. But, we can approximate $f(x)$ using other functions with simple analytic integrals, which may describe $f(x)$ better. With the trapezoidal rule, the function is approximated as a line from point a to point b, forming a trapezoid.



$$\int_a^b \mathrm{d}x\, f(x) \approx \frac{b-a}{2}\Big(f(a)+f(b)\Big) \tag{2}$$

Note that this method requires two function evaluations. Again, the integral can be divided into smaller intervals and the trapezoid rule can be performed for each division.

We can use even higher order polynomials for the approximating function, which really is just used to interpolate $f(x)$. For Simpson's rule, a second order polynomial is used.



$$\int_a^b \mathrm{d}x\, f(x) \approx \frac{b-a}{6}\left(f(a) + 4f\left(\tfrac{a+b}{2}\right) + f(b)\right) \tag{3}$$

This approximation requires three function evaluations. In general, these types of approximations are called the Newton-Cotes formulas. However, as the degree of the approximating polynomial becomes very high, the approximation can perform very poorly. This is because high order interpolating polynomials suffer from Runge's phenomenon, where the polynomial begins to fluctuate wildly. Despite this, any Newton-Cotes method can be used for composite integration.

When using composite integration methods, the accuracy and precision of the approximation can typically be improved by increasing the number of divisions. However, this can quickly become very computationally expensive. An alternative to performing a fixed number of divisions is to instead use adaptive division.

1. For each division the approximate integral is evaluated, using any numerical integration method.
2. This division is subdivided, typically in two. The approximate integral for each subdivision is calculated.
3. If the approximate integrals for the subdivisions, subtracted from the approximate integral for the division, is below some cut-off error, the subdividing process stops. Otherwise, subdivision continues with Step 2.
4. Terminate the algorithm when the error estimate for all divisions is below the cut-off error.

This method is called adaptive quadrature and can be applied to most methods for performing numerical integration. A cut-off error needs to be provided, and usually the maximum number of allowed subdivisions.

A similar method to adaptive quadrature also exists for Monte Carlo integration, importance sampling. A histogram of all the sampled points is kept, and this histogram is then used to sample new points. If a bin in the histogram contains a large number of points, then this bin will be sampled more often, while bins with a small number of points will be rarely sampled. The binning can be made adaptive, so that the bins with a large number of points can be divided into smaller bins. As more points are sampled, this histogram begins to provide an approximation of the distribution being sampled and consequently increases the convergence of the integration. There are a number of tricky details on how to implement importance sampling. Many of these issues have been solved with the VEGAS algorithm, one of the most commonly used Monte Carlo integration methods.

Goal 1 Implement a class that performs integration using the composite midpoint rule in one dimension. Check what the uncertainty is as a function of sampled points n for a number of functions that can be analytically integrated. Use the class to demonstrate integrating some functions which cannot be analytically integrated. Demonstrate convergence and perform timing tests.

Goal 2 Implement a class that performs integration using the composite trapezoidal rule in one dimension. Perform a similar analysis as for Goal 2. Compare and contrast with midpoint integration.

Goal 3 Implement a class that performs integration using the composite Simpson's rule in one dimension. Perform a similar analysis as for Goal 2. Compare and contrast with midpoint and trapezoid integration.

Goal 4 Implement a class that performs adaptive integration, using the integration methods from Goal 1, Goal 2, and Goal 3.

Goal 5 Implement a class that performs Monte Carlo integration, just as was done in Problem Set 2, except now for just one dimension. Compare to the results of Goal 1 through Goal 4.

Goal 6 Generalize the classes from Goal 1 to Goal 5 to n-dimensions. Analyse the results for up to four dimensions.

Goal 7 Implement an importance sampling Monte Carlo integration method, compare to the other integration methods for high dimension integrals.