Introduction

Determining the cross section of a particular interaction is desirable in multiple particle physics experiments, as the cross section is an indication of the probability of that particular interaction occurring (REFERENCE). Calculating the cross section of a two-to-two process, a common example being electron scattering, requires the integration over two variables; in general, a two-to-n process requires integrating over 3n-4 variables. As a typical LHC event produces hundreds of particles, evaluating the cross section would involve hundreds of variables. This is an impossible task to perform analytically, thus numerical integration methods are relied upon to perform the calculation.

Several numerical integrations are available that could be used for this task, each performing differently and with its own characteristics. Traditional quadrature-based methods, as the Newton-Cotes rules, are simple to implement and are deterministic in nature, allowing for a fast convergence as a function of sampling points. Random number-based methods, such as Monte Carlo integration, relies on randomly sampling the function to evaluate the integral, and therefore performs differently to quadrature methods. Each method could be enhanced by adapting which points are sampled to provide faster convergence. The MISER algorithm (REFERENCE) and the VEGAS algorithm (REFERENCE) are two examples of adaptive Monte Carlo integration, where the random number distribution is adapted over several iterations to converge faster in comparison to using a uniform distribution. Generalizing and implementing these methods to multiple dimensions would alter the characteristics of each method.

The aim of this project was to investigate and characterize multiple numerical integration methods and determine which would be best suited for use in particle physics. The integration methods investigated were the Newton-Cotes quadrates up to the second-degree polynomial, adaptive integration using Newton-Cotes, Monte Carlo integration, and its recursive stratified sampling based counterpart. The main objectives were to implement these functions in Python, generalize them to multiple dimensions, and characterize its performance by demonstrating convergence and evaluating its execution speed.

Physics Review

The Newton-Cotes quadrature rules are a set of rules used to calculate the integral by numerically interpolating the integrand as a polynomial through equally spaced intervals. The simplest of such rules is the midpoint rule, where the function is approximated as a 0th order polynomial evaluated at the midpoint of the limits a and b, as shown in FIGURE SOMETHING. The midpoint rule is mathematically expressed as

MIDPOINT RULE EQUATION.

The function could also be approximated as a 1st order polynomial, in which case the quadrature is based on determining the area of a trapezium. This rule is the trapezium rule and is illustrated in FIGURE SOMETHING. It is mathematicaly expressed as

TRAPEZIUM RULE EQUATION.

Simpson’s rule uses a 2nd order polynomial to approximate the function, theoretically increasing the accuracy of the value and increasing the speed of convergence. The rule is shown in FIGURE SOMETHING, with its mathematical expression being

SIMPSONS RULE EQUATOIN.

Higher order polynomial approximations are available and should theoretically increase the accuracy of the evaluation, however are rarely used due to the possibility of the approximation suffering from Runge’s Phenomenon. When approximating certain functions such as the Runge Function, the approximation would fluctuate wildly near the limits as the degree of the polynomial increases, as shown in FIGURE SOMETHING. Runge’s Phenomenon shows that using higher order polynomials are not always advantageous; therefore, only Newton-Cotes quadrature rules up to Simpson’s rule were analyzed for this project.

The Newton-Cotes methods could provide a better estimation by dividing the integral to a series of equally spaced intervals and applying the Newton-Cotes rule at each division. Generalizing this process to each method leads to the Newton-Cotes formula,

NEWTON COTES FORMULA,

where wi are the weights used for each point. The weights can be derived from the previous definitions of each rule, and are shown in table something (REFERENCE).

TABLE CONTAINING THE RULES FOR EACH THINGY.

The convergence of each method are well known and are shown in TABLE SOMETHING. Due to the higher order approximation of Simpson’s rule, it naturally converges the fastest at O(1/n^4). As the midpoint rule, and trapezium rule approximates the function as straight lines, the convergence of the two are of the same order O(1/n^2). Counterintuitively, the midpoint rule has a better bound compared to the trapezium rule, arising from how the trapezium rule systematically overestimates the value of the integral when the function has a positive curvature and underestimates the value at negative curvature (SEE APPENDIX FOR MORE INFORMATION).

The convergence of the Newton-Cotes quadrature rules could be improved using adaptive quadrature by systematically choosing the appropriate intervals to integrate over rather than equally spaced intervals. This process is done by first approximating the value at one division using a Newton-Cotes rule, after which the division is subdivided into two. The value of each subdivision is evaluated using the same Newton-Cotes rule. If the difference between the value of the division and the sum of the two subdivisions are less than an error tolerance, then the division stops and returns the value. Else, the two subdivisions are divided again, and the process repeats until the error tolerance condition is met. The selective choosing of intervals allows for a faster convergence and a better execution speed, as it would only perform the calculations at the most relevant sample points.

Newton-Cotes quadratures could be generalized to N dimensions by applying the quadrature rule in each dimension throughout all sampling points. The sampling points are found from the cartesian product of the sampling points required in each dimension. Summing over all sampling points lead to the general equation

NC MULTIDIMENSIONAL EQUATION,

The two dimensional case is illustrated in FIGURE SOMETHING, where the sampling points shown are the cartesian product of the sampling points P1xP2.

A problem develops when using Newton-Cotes quadratures in multiple dimensions. In FIGURE SOMETHING, 12 points were used for the sampling points in two dimensions, even when only 3 and 4 points were used in each dimension. The number of total sampling points used for quadrature increases exponentially as the number of dimensions increase, which decreases the performance of the method. The error scaling for each method is shown in TABLE SOMETHING.

Monte Carlo Integration is another numerical integration method, utilising random numbers to sample the integrand within the limits to determine its actual value. Monte Carlo takes advantage on the definition of the function’s average over an interval, using it to evaluate the integral as

MONTE CARLO INTEGRATION FORMULA

The method is easily generalizable to multiple dimensions, whereby the equation is given as

MONTE CARLO INTEGRATION FORMULA IN MULTIPLE DIMENSIONS

The convergence of Monte Carlo scales with O(1/√N) (SEE APPENDIX FOR DERIVATION) and is dimensionally invariant. This property is Monte Carlo’s most important characteristic, as it shows how the method compares against Newton-Cotes in one dimension and in multiple dimension. Monte Carlo is expected to converge slowly in one dimension in comparison with Newton-Cotes, as the deterministic nature requires only a few sampling points before it converges. However, in multiple dimensions, Newton-Cotes deteriorates in performance and is eventually surpassed by Monte Carlo methods, thereby making it a better choice for multidimensional problems.

The convergence of Monte Carlo methods can be improved by implementing variance reduction techniques to speed convergence. One such method is the recursive stratified sampling technique, which the MISER algorithm is based on (REFERENCE), and is not unlike adaptive quadrature. The integral is divided into several bins, and each bin is sampled using a few random numbers, to determine its value as well as its variance. The variance is calculated as

VARIANCE FORMULA

If the variance in a particular bin is larger than a tolerance, the bin is subdivided into two smaller bins and the process is repeated. Similarly to adaptive quadrature, this method has the effect of selectively choosing the intervals and using more intervals at areas where the variance is the largest. ILLUSTRATION NEEDED OR NA? This not only increases the convergence of the integral by using only the required number of sampling points, but it also increases the execution speed of the algorithm as well.

Algorithms and code structure

.