**How can you use quadratic polynomials to estimate the integral of a function?**

Simpson’s Rule

Approximate a function by a polynomial over an interval and estimate it’s integral that way.

Integral of f(x) over [a,b] is approximately integral of p2(x) over [a,b] where p2(x) is a polynomial that has nodes a, a+b/2, b

Approximation = 1/6 (b-a) ( f(a) + 4(a+b/2) + f(b))

Simpson’s Composite Rule

Approximate a function by a piecewise polynomial over an interval to estimate it’s integral

First subdivide interval into an even number N of intervals (N even)

Then approximate the integral by:

* h/3 (f(a) + f(b) + 4 sum of function values at nodes with odd indices + 2 sum of even nodes

Local Error = On one single subinterval

* Linear approximations (Trapezoidal) -- O(h^3)
* Quadratic approximations (Simpson’s) – O(h^5)

Global Error = On multiple stitched together intervals

* Quadratic = N \* O(h^5) = C / h \* O(h^5) = O(h^4)

Comparison

* Romberg Scheme requires the number of interval to be 2^N
* Simpson’s requires the number of intervals to be even

**How to generalize this approach for evaluating integrals?**

* Newton who did a lot of things, known for coming up with descriptions of motion and description of light, came up with a formula now called Newton’s Codes quadrature (means to measure the area, bounded by four lines)

Idea: Approximate f by a polynomial of degree n

F approximated by p in Pi\_n

The formula you end up with is:

Integral of f from a to b = integral of interpolating polynomial at n+1 nodes

= A\_0 f(x\_0) + A\_1 f(x\_1) + … + A\_n f(x\_n)

N = 1 🡪 Trapezoidal, N= 2 🡪 Simpson’s Rule

**Are there any different methods of evaluating the integral of a function?**

Gauss, who did a lot of things, Gaussian elimination, proving the central limit theorem, developing a theory of non-Euclidean geometry, also came up with an idea by questioning whether there needed to be a uniform partition of the function.

* Idea: consider the nodes x\_0, …, x\_n as variable, then you have 2n+2 parameters to determine, as A\_0, …, A\_n are variable
* Determine these unknowns (optimal nodes and their coefficients) by making the formula for the integral of exact for Pi\_2n+1
* Although this is a nonlinear equation and not guaranteed to have a solution, Gauss showed that there is a unique solution, with the optimal nodes, x\_0, .., x\_n are roots of certain orthogonal polynomials (nodes have been tabulated)

Examples:

* N = 1, integral f(x) over interval [-1, 1] = f(1/sqrt(3)) + f(-1/sqrt(3))

This approximation of the integral of a function of an interval is exact for polynomials up to degree 3

* N = 2, integral f(x) over interval [-1,1] = 5/9f(-sqrt(3/5) + …

This approximation of a function is exact for polynomials up to degree 5

If evaluating [a,b] you can scale nodes using a linear transformation

* N = 1, integral of 1/(1+x^2) dx from [0, 1] (which equals pi/4)
* Using the linear transformation u = 2x – 1 🡪 u in [-1,1], du = 2 dx
* I = integral of (1 + (1+u)/2)^-1 du = plug in 1/sqrt(3) and – 1/sqrt(3) in to function
* Can also do N = 2 by plugging into Gauss quadrature formula. So Gauss solved very specific problem of find formulas that approximate the integral of polynomials in p 2N + 1 over interval [-1,1] exactly
  + Integral [-1,1] = A\_0 f(x\_0) + A\_1 f(x\_1) for f in pi 3

Amazing, all you need to do is know the value of the function at 2 points and you can find the integral

* Furthermore the coefficients are always positive! This means you can always avoid subtractive cancellation, by either having a function that is always positive, or by shifting the function up until all of it is positive over the interval and then subtracting that amount.
* Error O(H^2n+1)

Finally you can try to find nodes such that the coefficients A\_0, …, A\_1 of the function values are 1. There are tons and tons of other methods.

**What about multivariate generalizations?**

* Because of the curse of dimensionality, sampling functions in meshes requires N^d, where d is the number of dimensions you are integrating over
* Instead what is used is Monte Carlo method, you pick points at random

**What about choose interpolation nodes based on the properties of the function?**

* If you have shocks, a function suddenly decreases, or increases suddenly, then you want to condense interpolation nodes around those shocks
* Nodes are selected locally based on local variation of f