Gaussian Quadrature II and Adaptive Quadrature

Friday, November 8, 2024 10:38 AM

Class 32: November 8, 2024

Recall: Last time, we introduced a way to compute quadratures of the highest order possible (for n+1 nodes and n+1 weights) by writing down equations so that the rule is exact for polynomials up to degree <= 2n+1 (2n+2 degrees of freedom). The first way we did this was to write down equations for the rule being exact for x^j, for j= 0,1,…,2n+1:

\Int_a^b x^j dx = w0 x0^j + w1 x1^j + … + wn xn^j

This gives us a **non-linear system** of 2n+2 equations in 2n+2 variables. It has a unique solution (up to re-labeling nodes or weights). For [a,b] = [-1,1], this gives us the "n+1 point Gauss Legendre" quadratures. We mentioned the first two: midpoint (w0=2, x0=0), which is O(h^3) for one interval and O(h^2) for composite, and 2-point (w0=w1= 1, x0=-1/sqrt(3), x1=1/sqrt(3)), which is O(h^5) for one interval and O(h^4) for composite.

We then proved a very important result: The nodes for the n+1 point Gauss-Legendre quadrature are the unique n+1 zeroes of the n+1 Legendre polynomial P_{n+1}(x). We could, once we have the zeroes, solve a system of n+1 linear equations for the weights OR compute them as the integrals of the Lagrange polynomials, but orthogonal polynomials come to the rescue here as well: there is a formula for the weights! To summarize:

P_{n+1}(x_j) = 0 ;
w_j = 2 / ((1-x_j^2) [P'_{n+1}(x_j)]^2)

Finally, we mentioned that families of polynomials for different weights w(x) are, in fact, useful to generate Gaussian quadratures which are used in probability and singular integration, to name two broad fields of application.

Today, we will wrap up Gaussian quadrature rules and will introduce the concept of Adaptive Quadrature (which you will get to work with on your next lab!).

Mathematical reasoning is identical using

$$\langle f, g \rangle_\omega = \int_a^b f(x) g(x) \omega(x) dx .$$

→ Build family $\{\Phi_j\}_{j=0}^{\infty}$

↳ 3 term recursion. → Other formula (e.g $\Phi_j'(x)$)

1. Find nodes by $\Phi_{n+1}(x_j) = 0$.

   → Check if they are known / read from file.

   → Use Newton w good initial guess to find these as roots.

   → Golub-Welsch algo → x_j are

→ Golub-Welsch algo → x_j are eigenvalues of a $n+1 \times n+1$ tri.-diagonal matrix!

Compute weights using formula:

$$\left\{ w_j = \frac{a_{n+1}}{a_n} \frac{\int_a^b \bar{\Phi}_n^2(x)\, w(x)\, dx}{\bar{\Phi}_{n+1}'(x_j)\, \bar{\Phi}_n(x_j)} \right\}$$

ERROR ESTIMATES

Assumption: $f \in C^{(2n+2)}([a,b])$, $\exists\, \eta \in (a,b)$ s.t.

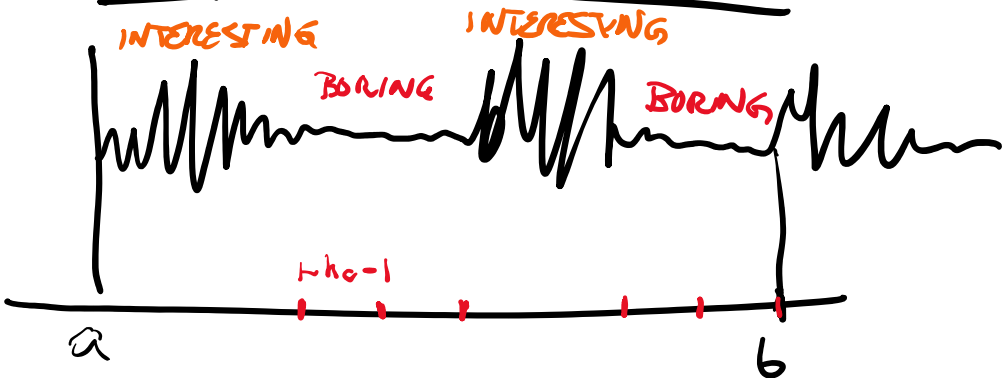
$$Err = \frac{f^{(2n+2)}(\eta)}{(2n+2)!} \left(\int_a^b \psi(x)^2 w(x)\, dx \right)$$

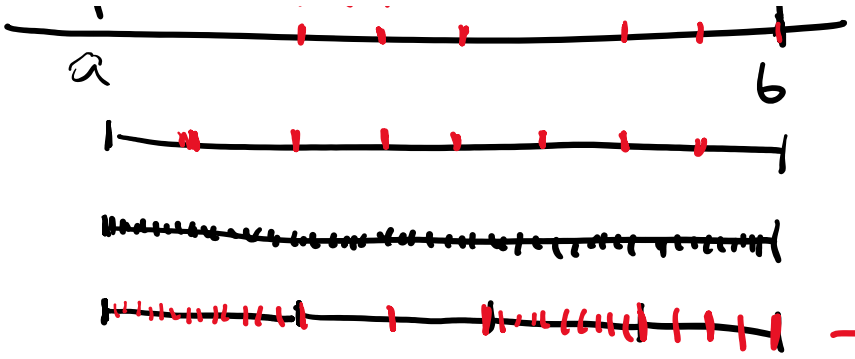
$$\psi(x) = \prod_{j=0}^{n} (x - x_j)$$

$$O\left(\left[\frac{b-a}{N}\right]^{2n+3}\right)$$

COMP RULE $\left(\frac{b-a}{N}\right)^{2n+2}$.

Adaptive Quadrature





Sketch:

0 Pick a composite quadrature Q.

1 Apply $Q_{[a,b]}$ (one interval rule) and compare with $Q_{\{\overset{I_1}{[a,m]},\overset{I_2}{[m,b]}\}}$

$m = \frac{a+b}{2}$.

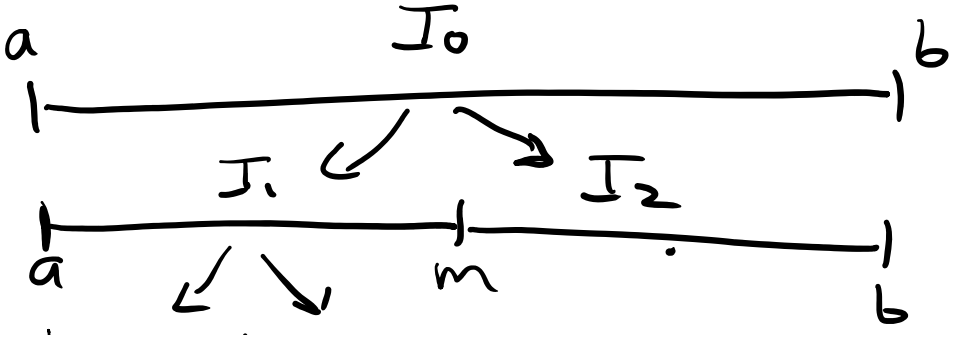
Look at

$$Q_{[a,b]} - Q_{\{I_1,I_2\}}$$

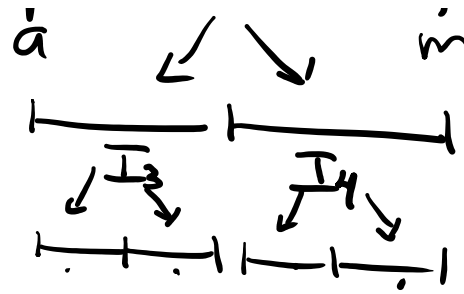
$$= \left(Q_{[a,b]} - \int_a^b f(x)dx\right) - \left(\int_a^b f(x)dx - Q_{\{I_1,I_2\}}\right)$$

→ Estimate $\propto$ Err for $Q_{\{I_1,I_2\}}$.

Graphical Rep of algorithm:



Est: $Q_{I_0} - (Q_{\{I_1, I_2\}})$



$-(I_{1,3} I_{2,3})$

Recursive and Non-recursive.

↳ Stability

↳ Parallelization.