Conditioning

$$\int_a^b f(x)\,dx \approx I = \int_a^b \hat{f}(x)\,dx$$

Derive the (absolute) condition number for numerical integration.

$$\hat{f}(x) = f(x) + \underbrace{e(x)}_{\text{perturbation}}$$

$$\left| \int_a^b f(x)\,dx - \int_a^b \hat{f}(x)\,dx \right| = \left| \int_a^b e(x)\,dx \right|$$

$$\leq \int_a^b |e(x)|\,dx \leq \int_a^b \max_{x\in[a,b]} |e(x)|\,dx = (b-a)\cdot \max_{[a,b]} |e(x)|$$

Interpolatory Quadrature

Design a quadrature method based on interpolation.

$$f(x) \approx \sum \alpha_i \varphi_i(x)$$

$$\Rightarrow \int_a^b f(x)\,dx \approx \int_a^b \sum \alpha_i \varphi_i(x)$$

- linear.
- interpolatory quadrature.

Interpolatory Quadrature: Examples

$$f(x) \approx \sum_i f(x_i) \underbrace{l_i(x)}$$

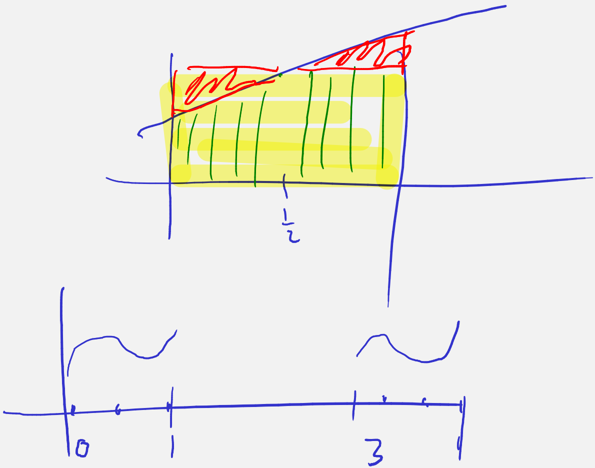
↳ Lagrange polynomial.

$$\prod_{i \neq j} \frac{(x - x_j)}{(x_i - x_j)}$$

$$\int_a^b f(x)\,dx \approx \sum f(x_i) \underbrace{\int_a^b l_i(x)\,dx}_{w_i \text{: weights}}$$

- x_i: equi-spaced : Newton-Cotes
- x_i Chebyshev : Clenshaw-Curtis.

Interpolatory Quadrature: Examples



Interpolatory Quadrature: Computing Weights $\int f(x)dx \approx \sum f(x_i)\, w_i$

How do the weights in interpolatory quadrature get computed?

$$(b-a) = \int_a^b 1\, dx = 1\cdot w_1 + \cdots + 1\cdot w_n$$

$$\frac{1}{2}(b^2-a^2) = \int_a^b x\, dx = x_1\cdot w_1 + \cdots + x_n w_n$$

$$\vdots$$

$$\frac{1}{k+1}\left(b^{k+1}-a^{k+1}\right) = \int_a^b x^k = x_1^k\cdot w_1 + \cdots + x_n^k \cdot w_n$$

"method of the undetermined $V^T \vec{w}$

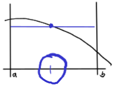
coefficients

Demo: Newton-Cotes weight finder [cleared]

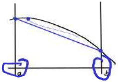
Examples and Exactness

To what polynomial degree are the following rules exact?

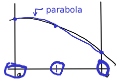
Midpoint rule $\quad (b-a)f\left(\frac{a+b}{2}\right)$



Trapezoidal rule $\quad \frac{b-a}{2}(f(a)+f(b))$



Simpson's rule $\quad \frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right)$



Midpoint: deserves 0, gets 1 (linears)

Trapezoidal: 1 (linears)

Simpson's: technically 2 (quadratics), actually cubics (3)

can use diff of these as error estimate "a-posteriori"

Interpolatory Quadrature: Accuracy

"a priori"

Let p_{n-1} be an interpolant of f at nodes $x_1, \ldots, x_n$ (of degree $n-1$)

Recall

$$\to \quad \sum_i \omega_i f(x_i) = \int_a^b p_{n-1}(x)\mathrm{d}x. \qquad \|f - p_n\|_\infty \le C \cdot h^n \|f^{(n)}\|_\infty$$

What can you say about the accuracy of the method?

$$\left| \int_a^b f(x)\,dx - \int_a^b p_{n-1}(x)\,dx \right|$$

$$\le \int_a^b \left| f(x) - p_{n-1}(x) \right| dx$$

$$\le (b-a) \cdot \underbrace{\max_{x \in [a,b]} \left| f(x) - p_{n-1}(x) \right|}$$

$$\le C(b-a)\, h^n \|f^{(n)}\|_\infty = C h^{n+1} \|f^{(n)}\|_\infty$$

Quadrature: Overview of Rules

	n	Deg.	Ex.Int.Deg. (w/odd)	Intp.Ord.	Quad.Ord. (regular)	Quad.Ord. (w/odd)
		$n-1$	$(n-1)+1_{\text{odd}}$	n	$n+1$	$(n+1)+1_{\text{odd}}$
Midp.	1	0	1	1	2	3
Trapz.	2	1	1	2	3	3
Simps.	3	2	3	3	4	5
—	4	3	3	4	5	5

- n: number of points
- "Deg.": Degree of polynomial used in interpolation ($= n-1$)
- "Ex.Int.Deg.": Polynomials of up to (and including) this degree *actually* get integrated exactly. (including the odd-order bump)
- "Intp.Ord.": Order of Accuracy of Interpolation: $O(h^n)$
- "Quad.Ord. (regular)": Order of accuracy for quadrature predicted by the error result above: $O(h^{n+1})$
- "Quad.Ord. (w/odd):" Actual order of accuracy for quadrature given 'bonus' degrees for rules with odd point count

Observation: Quadrature gets (at least) 'one order higher' than interpolation–even more for odd-order rules. (i.e. more accurate)

Interpolatory Quadrature: Stability

$|Err| \le |b-a| \cdot \max_{[a,b]} |e(x)|$

Let p_n be an interpolant of f at nodes $x_1, \ldots, x_n$ (of degree $n-1$)
Recall

$$\sum_i \omega_i f(x_i) = \int_a^b p_n(x)\mathrm{d}x$$

What can you say about the stability of this method?

$\hat{f}(x) = f(x) + e(x)$

$\left| \sum_i w_i f(x_i) - \sum_i w_i \hat{f}(x_i) \right| \le \sum_i |w_i \underline{e(x_i)}|$

$\le \left(\sum_i |w_i|\right) \cdot \|e\|_\infty$

$\underline{\sum_i w_i = |b-a|}$

: negative weights bad!

About Newton-Cotes

What's not to like about Newton-Cotes quadrature?

Gaussian Quadrature

So far: nodes chosen from outside.
Can we gain something if we let the quadrature rule choose the nodes, too? Hope: More design freedom $\to$ Exact to higher degree.

Idea: Let the method choose both w_i, x_i $\to$ $2n$ unknowns

$\Rightarrow$ Hopefully $2n$ polynomials can be integrated exactly

$\to$ optimization $\to$ clunky $\to$ avoid?

Demo: Gaussian quadrature weight finder [cleared]

Composite Quadrature

High-order polynomial interpolation requires a high degree of smoothness of the function.
Idea: Stitch together multiple lower-order quadrature rules to alleviate smoothness requirement.

e.g. trapezoidal