Logistics:

- HW 13 (due Wed)
- 4CH2 (due Dec 14)
- Final

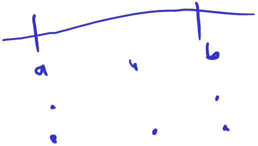
Review:

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

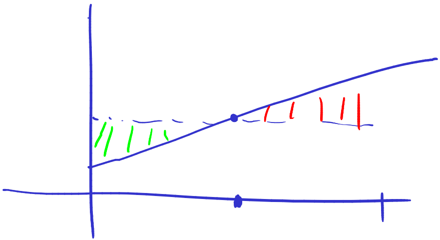
$f=1$: $w_1 + w_2 + \cdots + w_n = b-a$

$f(x)=x$: $w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = \frac{b^2}{2} - \frac{a^2}{2}$

$$V^T \vec{w} = \begin{pmatrix} \int_a^b 1 \\ \int_a^b x \\ \\ \int_a^b x^{n-1} \end{pmatrix}$$



midpoint
Newton-Cotes quad rules
trapezoidal
Simpson's rule



odd # points integrate an extra degree

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
S. 3/8	4	3	3	4	5	5

$E(n) = O(h^{?})$

- 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

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?

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

$\left| \sum_{i=1}^{n} w_i f(x_i) - \sum w_i \hat{f}(x_i) \right| \le \sum_{i=1}^{n} |w_i e(x_i)| \le \|e\|_\infty \sum_{i=1}^{n} |w_i|$

So, what quadrature weights make for bad stability bounds?

$\sum_{i=1}^{n} w_i = b-a$

weights w/ oscillating signs bad

About Newton-Cotes

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

Demo: Newton-Cotes weight finder [cleared] (again, with many nodes)

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 → Exact to higher degree.

Idea: still method of undet. coeffs
But: "nodes also unknown"

→ analytically solvable
solution for nodes is "roots of Legendre polys"

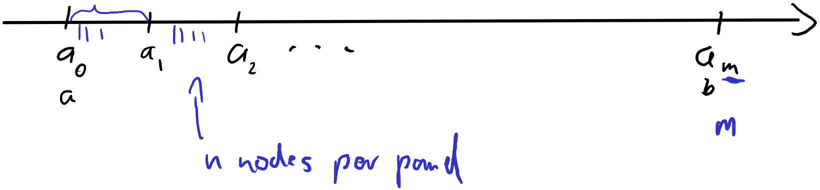
Demo: Gaussian quadrature weight finder [cleared]

⊕ Radau
⊕ Lobatto

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



Error in Composite Quadrature

$a_{i+1} - a_i = h$

What can we say about the error in the case of composite quadrature?

$+1$ in n odd

Error for single interval: $\left| \int_{a_i}^{a_{i+1}} f - p_{n-1}\, dx \right| \le C \cdot h^{n+1} \cdot \|f^{(n)}\|_\infty$

$$\left| \int_a^b f(x)\,dx - \sum_{j=1}^{m} \sum_{i=1}^{n} \omega_{j,i} f(x_{i,j}) \right| \le C \|f^{(n)}\|_\infty \sum_{j=1}^{m} (\underbrace{a_j - a_{j-1}}_{h})^{n+1}$$

panels

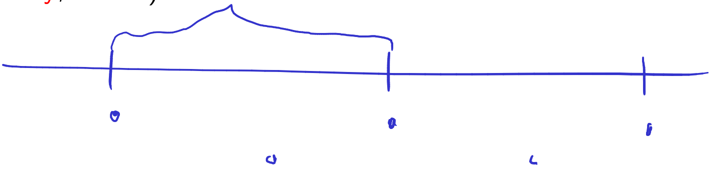
quad rule on panel m

$$\le C \cdot \|f^{(n)}\|_\infty \sum_{j=1}^{m} \underbrace{(a_j - a_{j-1})^n}_{\le h^n} (a_j - a_{j-1}) \le C \cdot \|f^{(n)}\|_\infty h^n (b-a)$$

Composite Quadrature: Notes

Observation: Composite quadrature loses an order compared to non-composite.

Idea: If we can estimate errors on each subinterval, we can shrink (e.g. by splitting in half) only those contributing the most to the error. (adaptivity, $\rightarrow$ hw)



Taking Derivatives Numerically

Why *shouldn't* you take derivatives numerically?

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

- Cancellation
- Susceptible to noise
- ∂_x is an unbounded operator

$\|f\|_\infty \le 1 \not\Rightarrow \|f'\|_\infty$ bounded

$e^{i\alpha x}$ $\qquad (e^{i\alpha x})' = (i\alpha) e^{i\alpha x}$

unbounded

Numerical Differentiation: How?

How can we take derivatives numerically?

$$f(\xi) \approx p_{n-1}(\xi) = \sum_{i=1}^{n} \alpha_i \varphi_i(\xi)$$

$$f'(\xi) \approx p_{n-1}'(\xi) = \sum_{i=1}^{n} \alpha_i \varphi_i'(\xi)$$

Demo: Taking Derivatives with Vandermonde Matrices [cleared] (Basics)

Numerical Differentiation: Accuracy

How accurate is numerical differentiation (with a polynomial basis)?

$$f(x) - p_{n-1}(x) = \frac{f^{(n)}(\xi)}{n!} \prod_{i=1}^{n} (x - x_i)$$

$$f'(x) - p'_{n-1}(x) \approx \frac{f^{(n)}(\xi)}{n!} \overbrace{\left(\prod_{i=1}^{n} (x - x_i)\right)'}^{\le h^{n-1}}$$

Ignoring ξ depending on x

n points:
- Derivative h^{n-1}
- Interpolation: h^n
- Quadrature h^{n+1} or h^{n+2}

Demo: Taking Derivatives with Vandermonde Matrices [cleared]
(Accuracy)

Differentiation Matrices

How can numerical differentiation be cast as a matrix-vector operation?

Demo: Taking Derivatives with Vandermonde Matrices [cleared] (Build D)