	n	Deg. (reg.)	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
unnamed	4	3	3	4	5	5

- n: number of points
- "Deg. (reg.)": Degree of polynomial used in interpolation to build the quadrature rule. $(=n-1)$
- "Ex.Int.Deg.": Polynomials of up to (and including) this degree *actually* get integrated exactly. (including the odd-order bump) $\left(=\begin{cases} n-1 & \text{even} \\ n & \text{odd} \end{cases}\right)$
- "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 $\left(\begin{cases} O(h^{n+1}) & \text{even} \\ O(h^{n+2}) & \text{odd} \end{cases} \right)$

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

8.1.4 Gaussian Quadrature

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

Demo: Gaussian quadrature weight finder

Exact for polynomials up to degree $2n-1$

↳ Setup the same as interpolatory q.

But with roots of Legendre poly. as nodes

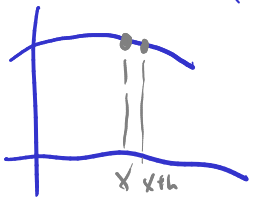
↳ Have a right to expect exactness upto degree $n-1$

↳ Order of accuracy: $2n$

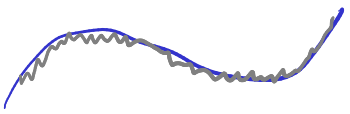
8.2 Numerical Differentiation

Taking Derivatives Numerically

- Why *shouldn't* you take derivatives numerically?
  - ill-conditioned - because it has a null space
    - $(\sin(\alpha x))' = \alpha \cos(\alpha x)$ $D \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$
    - ↑ unbounded $D \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$
  - $f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$



Catastrophic cancellation

-  : Amplifies noise

- Interpolation with n points $\quad \varepsilon_{interp}(h) \simeq O(h^n)$

  Num. diff. with n points $\quad \varepsilon_{diff}(h) \simeq O(h^{n-1})$

$$f(x_i) = y_i$$

$$\hat{f}(x) = \sum_i \alpha_i \varphi_i(x)$$

$$\hat{f}'(x) = \sum_i \alpha_i \varphi_i'(x)$$

Demo: Taking Derivatives with Vandermonde matrices

Finite Differences

- If you *absolutely* have to take num. derivatives, what could you do?
  - Compute interpolation coefficients, differentiate basis
  - 'Finite Differences'

Demo: Finite Differences vs Noise
Demo: Floating point vs Finite Differences