<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Midp.</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Trapz.</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Simps.</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>unnamed</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

- \( n \): 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) \((=\begin{cases} n-1 & \text{even} \\ n & \text{odd} \end{cases})\)
- “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
\[
\begin{cases} 
O(h^{n+1}) & \text{even} \\
O(h^{n+2}) & \text{odd}
\end{cases}
\]

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 $\rightarrow$ 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 up to 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
  - \((\sin(\alpha x))' = \alpha \cos(\alpha x)\)
  - Unbounded
  - \(f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}\)
  - Catastrophic cancellation
  - Amplifies noise
Interpolation with $n$ points

$E_{\text{interp}}(h) = O(h^n)$

Numerical differentiation with $n$ points

$E_{\text{diff}}(h) = O(h^{n-1})$

$f(x_i) = y_i$

$\hat{f}'(x) = \sum \frac{1}{h} \left( f(x_i) - f(x_{i-1}) \right)$

$\hat{f}''(x) = \sum \frac{1}{h^2} \left( f(x_{i+1}) - 2f(x_i) + f(x_{i-1}) \right)$
Demo: Taking Derivatives with Vandermonde matrices
Finite Differences

- If you *absolutely* have to take numerical derivatives, what could you do?
  - Compute interpolation coefficients, differentiate basis
  - ‘Finite Differences’
Demo: Finite Differences vs Noise
Demo: Floating point vs Finite Differences