Logistics:
- HW 13 (due Wed)
- CCH12 (due Dec 14)
- Final

Review:

\[ \int_a^b \sum_{i=1}^n f(x_i) dx \approx \sum_{i=1}^n u_i \cdot f(x_i) \]

\[ \phi_i \cdot \psi(x) ; w_1 + w_2 + \cdots + w_n = b - a \]

\[ \phi(x) \cdot \psi(x) ; w_1 x_1 + w_2 x_1^2 + \cdots + w_n x_n = \frac{b^n}{2} - \frac{a^n}{2} \]

V=W^T

midpoint Newton-Cotes quadrature
odd # points integrate an extra degree
## Quadrature: Overview of Rules

<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>$(n-1)+1_{\text{odd}}$</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>S. 3/8</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.”: 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

$$\sum_i \omega_i f(x_i) = \int_a^b p_n(x)dx$$

What can you say about the stability of this method?

So, what quadrature weights make for bad stability bounds?

$$\sum_i \omega_i = b-a$$

Weights with 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 $\rightarrow$ Exact to higher degree.

**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

\[ a \quad a_1 \quad a_2 \quad \ldots \quad a_m \]

\( n \) nodes per panel
What can we say about the error in the case of composite quadrature?

**Error for single interval:**

\[
\left| \int_a^b f(x) \, dx - \sum_{i=1}^{\infty} \sum_{j=1}^{n} w_{ij} f(x_{ij}) \right| \leq C \|f^{(n)}\|_{\infty} \sum_{j=1}^{n} (a_j - a_{j-1})^n (b - a)
\]

\[
\leq C \|f^{(n)}\|_{\infty} \sum_{j=1}^{n} (a_j - a_{j-1})^n (b - a) \leq C \|f^{(n)}\|_{\infty} h^n (b-a)
\]

\[a_{i+1} - a_i = h\]
**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
- \( \partial_x \) is an unbounded operator

\[\|f\|_1 \Rightarrow \|f'\|_1 \text{ bounded}\]

\[e^{iax} \quad (e^{iax})' = (ia)e^{iax}\]

unbounded
Numerical Differentiation: How?

How can we take derivatives numerically?

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

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

\[
\frac{f(x) - p_n(x)}{x - x_i} = \frac{f^{(n+1)}(\xi)}{n!} \prod_{i=1}^{n} (x - x_i)
\]

Ignoring \( f \) depending on \( x \)

**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)