<table>
<thead>
<tr>
<th>Today:</th>
<th>Announcements</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Calculus</td>
<td>- HW 11 (today!)</td>
</tr>
</tbody>
</table>
<pre><code>                                  | - Exam #3                      |
                                  | - Exam #4                      |
</code></pre>
Numerical Integration: About the Problem

What is numerical integration? (Or quadrature?)

\[ \int_{a}^{b} f(x) \, dx \quad \text{given } a, b, f \]

What about existence and uniqueness?

- Riemann integrability
- Lebesgue integrability

\[ \exists \text{ & uniqueness if } f \text{ is continuous and bounded} \]

\[ \lim_{n \to \infty} \sum fn = \lim_{n \to \infty} S fn \]
Derive the (absolute) condition number for numerical integration.
Interpolatory Quadrature

\[ \tilde{p} = p(\tilde{x}) \]

Design a quadrature method based on interpolation.

\[ \int_a^b f(x) \, dx \approx \sum_{i=0}^{n-1} \alpha_i \phi_i(x_i) \]

\[ \tilde{e} : \tilde{e}_i = \int_a^b \phi_i(x) \, dx \]

\[ V_2^2 = \tilde{p} \]

\[ \tilde{x} = v^{-1} \tilde{p} \]
\[
\sum_{a} \nu(x) \alpha x = \sum_{a} \nu_{a-1}(x) \alpha x = \sum_{1}^{\nu - 1} \alpha \sum_{a} \eta_{a}(x) = \alpha \cdot t
\]

\[
= \tilde{e}^T \tilde{\omega} - \tilde{e}^T \tilde{\gamma} \tilde{\omega} = \tilde{\omega} \cdot \tilde{\gamma}.
\]
Interpolatory Quadrature: Examples

- w/ Chebyshev nodes & Chebyshev polys
  - Chebyshev
  - Clenshaw-Curtis quadrature

- w/ equispaced nodes and monomials
  - Newton-Cotes quadrature
Interpolatory Quadrature: Computing Weights

How do the weights in interpolatory quadrature get computed?

\[ \int_a^b \varphi_i(x) \, dx = \mathbf{w} \cdot \varphi_i (\bar{x}) \quad i = 0, \ldots, n-1 \]

For monomials

\[ b-a = \int_a^b 1 \, dx = \omega_0 \cdot 1 + \omega_1 \cdot 1 + \omega_2 \cdot 1 + \cdots + \omega_n \cdot 1 \]

\[ \frac{1}{2} (b^2 - a^2) = \int_a^b x \, dx = \omega_0 \cdot x + \omega_1 \cdot x_1 + \omega_2 \cdot x_2 + \cdots + \omega_n \cdot x_n \]

\[ \frac{1}{k} (b^{k+1} - a^{k+1}) = \int_a^b x^k \, dx = \omega_0 \cdot x_0^k + \omega_1 \cdot x_1^k + \cdots + \omega_n \cdot x_n^k \]

\[ \mathbf{w} \quad \Rightarrow \quad \mathbf{V}^T \mathbf{w} \]

**Demo:** Newton-Cotes weight finder
\[ \int_0^1 p(x) \, dx = \frac{1}{2} \cdot f(0) + \frac{1}{2} \cdot f(1) \]

Midpoint rule and linear.
\[
\sum_{x} \rho(x) a_x = \sum_{x} \frac{1}{2} a_x
\]
Examples and Exactness

To what polynomial degree are the following rules exact?

- **Midpoint rule** \((b - a)f\left(\frac{a+b}{2}\right)\)
- **Trapezoidal rule** \(\frac{b-a}{2}(f(a) + f(b))\)
- **Simpson’s rule** \(\frac{b-a}{6} \left( f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right)\)
Interpolatory Quadrature: Accuracy

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

$$\sum_i \omega_i f(x_i) = \int_a^b p_{n-1}(x) dx.$$ 

What can you say about the accuracy of the method?
\[ \left| p_{n-1}(x) - f(x) \right| \leq C \cdot |f^{(n)}(x)| h^n \]

\[ \left| \int_a^b g(x) \, dx - \int_a^b p_{n-1}(x) \, dx \right| \leq \int_a^b |g(x) - p_{n-1}(x)| \, dx \]

\[ \leq \int_a^b |g(x) - p_{n-1}(x)| \, dx \]

\[ \leq \int_a^b \left( C \cdot \max_{\gamma \in [a,b]} |g^{(n)}(\gamma)| \right) \cdot h^n \, dx \]

\[ = C (b-a) \cdot \max_{\gamma \in [a,b]} \left| g^{(n)}(\gamma) \right| \cdot h^n \]

\[ \leq C \cdot h^{n+1} \]
# 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>$n - 1$</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Trapz.</td>
<td>2</td>
<td>1</td>
<td>$(n-1) + 1_{\text{odd}}$</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Simps.</td>
<td>3</td>
<td>2</td>
<td>$n - 1$</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3</td>
<td>$n - 1$</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

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

What can you say about the stability of this method?

$$\left| \sum \omega_i f(x_i) - \hat{f}(x) \right| = \left| \sum \omega_i e(x_i) \right| \leq \max_j |e(x_j)|$$
bad quadrature rule: oscillating weights

good quadrature: pos. weights

\[ \int_{-\frac{1}{2}}^{\frac{1}{2}} \max |w_i| (13) \leq 1 \]
What’s not to like about Newton-Cotes quadrature?
Composite Quadrature  \(\text{(not yet discussed)}\)

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

\[
\begin{array}{cccccc}
a_0 & a_1 & a_2 & \cdots & a_n \\
\alpha & & & & \beta \\
\end{array}
\]
Error in Composite Quadrature

(not yet discussed)

What can we say about the error in the case of composite quadrature?
**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, → hw)
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
In-Class Activity: Quadrature
Taking Derivatives Numerically

Why *shouldn’t* you take derivatives numerically?

**Demo:** Taking Derivatives with Vandermonde Matrices