CS 450: Numerical Analysis\textsuperscript{1}
Numerical Integration and Differentiation

University of Illinois at Urbana-Champaign

\textsuperscript{1}These slides have been drafted by Edgar Solomonik as lecture templates and supplementary material for the book “Scientific Computing: An Introductory Survey” by Michael T. Heath (slides).
\[ f(x) \approx \langle w, y \rangle \]

Newton-Cotes are exact for polynomials of degree \( n-1 \), when using \( n \) nodes.

ill-conditioned for large \( n \)

\( \to \) Chebyshev nodes (Clenshaw-Curtis quadrature)

Gaussian Quadrature (select both nodes and weights)

\( \to \) degree \( 2n-1 \)
piecewise interpolation → composite quadrature rules

progressive quadrature (rule for 2n nodes includes nodes used in rule for n nodes)

error scales as $O(h^n)$
Integrability and Sensitivity

- Seek to compute $\mathcal{I}(f) = \int_a^b f(x)dx$:

- The condition number of integration is bounded by the distance $b - a$: 
Quadrature Rules

- Approximate the integral $\mathcal{I}(f)$ by a weighted sum of function values:

- For a fixed set of $n$ nodes, polynomial interpolation followed by integration give $(n - 1)$-degree quadrature rule:
Determining Weights for Quadrature Rules

- A quadrature rule provides $x$ and $w$ so as to approximate

- Method of undetermined coefficients obtains $y$ from moment equations, which insure the quadrature rule is exact for all monomials of degree $n - 1$: 
Newton-Cotes Quadrature

- **Newton-Cotes** quadrature rules are defined by equispaced nodes on \([a, b]\):

- The *midpoint rule* is the \(n = 1\) open Newton-Cotes rule:

- The *trapezoid rule* is the \(n = 2\) closed Newton-Cotes rule:

- *Simpson’s rule* is the \(n = 3\) closed Newton-Cotes rule:
Consider the Taylor expansion of \( f \) about the midpoint of the integration interval \( m = (a + b)/2 \):

Integrating the Taylor approximation of \( f \), we note that the odd terms drop:
Error Estimation

- The trapezoid rule is also first degree, despite using higher-degree polynomial interpolant approximation, since

- The above derivation allows us to obtain an error approximation via a difference of midpoint and trapezoidal rules:
Error in Polynomial Quadrature Rules

- We can bound the error for an arbitrary polynomial quadrature rule by
Conditioning of Newton-Cotes Quadrature

- We can ascertain stability of quadrature rules, by considering the amplification of a perturbation $\hat{f} = f + \delta f$:

- Newton-Cotes quadrature rules have at least one negative weight for any $n \geq 11$: 
To obtain a more stable quadrature rule, we need to ensure the integrated interpolant is well-behaved as $n$ increases:
Gaussian Quadrature

So far, we have only considered quadrature rules based on a fixed set of nodes, but we may also be able to choose nodes to maximize accuracy:

The unique $n$-point Gaussian quadrature rule is defined by the solution of the nonlinear form of the moment equations in terms of both $x$ and $\omega$:
Using Gaussian Quadrature Rules

- Gaussian quadrature rules are hard to compute, but can be enumerated for a fixed interval, e.g. \( a = 0, b = 1 \), so it suffices to transform the integral to \([0, 1]\).

- Gaussian quadrature rules are accurate and stable but not progressive (nodes cannot be reused to obtain higher-degree approximation):
Progressive Gaussian-like Quadrature Rules

- *Kronod* quadrature rules construct \((2n + 1)-\)point \((3n + 1)-\)degree quadrature \(K_{2n+1}\) that is progressive with respect to Gaussian quadrature rule \(G_n\):

- *Patterson* quadrature rules use \(2n + 2\) more points to extend \((2n + 1)-\)point Kronod rule to degree \(6n + 4\), while reusing all \(2n + 1\) points.

- Gaussian quadrature rules are in general open, but *Gauss-Radau* and *Gauss-Lobatto* rules permit including end-points:
Composite and Adaptive Quadrature

- Composite quadrature rules are obtained by integrating a piecewise interpolant of $f$:

- Composite quadrature can be done with adaptive refinement:
More Complicated Integration Problems

- To handle improper integrals can either transform integral to get rid of infinite limit or use appropriate open quadrature rules.

- Double integrals can simply be computed by successive 1-D integration.

- High-dimensional integration is often effectively done by Monte Carlo:
Integral Equations

- Rather than evaluating an integral, in solving an *integral equation* we seek to compute the integrand. A typical linear integral equation has the form

\[ \int_a^b K(s, t)u(t)\,dt = f(s), \]  

where \( K \) and \( f \) are known.

- Using a quadrature rule with weights \( w_1, \ldots, w_n \) and nodes \( t_1, \ldots, t_n \) obtain
Numerical Differentiation

- Automatic (symbolic) differentiation is a surprisingly viable option:

- Numerical differentiation can be done by interpolation or finite differencing:
Accuracy of Finite Differences

- **Forward and backward differencing** provide first-order accuracy:

  \[
  f(x + h) = f(x) + h f'(x) + \frac{h^2}{2} f''(x) + \ldots
  \]

  \[
  f'(x) = \frac{f(x + h) - f(x)}{h} - \frac{h}{2} f''(x) + \ldots
  \]

  *Forward difference*

- **Centered differencing** provides second-order accuracy:

  \[
  f(x - h) = f(x) - h f'(x) + \frac{h^2}{2} f''(x) + \ldots
  \]

  \[
  f'(x) = \frac{f(x + h) - f(x - h)}{2h} + \frac{h}{2} f''(x) + \ldots
  \]

  \[
  f'(x) = \frac{f(x + h) - f(x - h)}{2h} + O(h^2)
  \]
Extrapolation Techniques

Given a series of approximate solutions produced by an iterative procedure, a more accurate approximation may be obtained by *extrapolating* this series.

\[ F(h) = a_0 + (a_1 h^p) + O(h^n) \]

In particular, given two guesses, *Richardson extrapolation* eliminates the leading order error term.

\[ F(h) = a_0 + (a_1 h^p) + O(h^n) \]
\[ F(h/2) = a_0 + (a_1 h^p) + O(h^n) \]

**Demo:** Richardson with Finite Differences

**Activity:** Richardson Extrapolation
\[ a_0 = f(h) - \frac{f(h) - f(h/2)}{\frac{1}{2^p}(1 - \frac{1}{2^p})} + o(h) \]

\[ = \frac{1 - (1 - \frac{1}{2^p})}{1 - \frac{1}{2^p}} \cdot f(h) + \frac{f(h/2)}{1 - \frac{1}{2^p}} \cdot o(h) \]
High-Order Extrapolation

- Given a series of $k$ approximations, Romberg integration applies $(k - 1)$-levels of Richardson extrapolation.

Extrapolation can be used within an iterative procedure at each step:

Steffensen's method for nonlinear solve (alternative to Newton and Secant)

$$x_{n+1} = x_n + \frac{f(x_n)}{1 - (f(x_n + f(x_n))/f(x_n))}$$

quadratic convergence