CS 450: Numerical Analysis
Numerical Integration and Differentiation

University of Illinois at Urbana-Champaign

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).
Integrability and Sensitivity

- Seek to compute $\mathcal{I}(f) = \int_a^b f(x) \, dx$:
  - $f$ is integrable if continuous and bounded.
  - Finite number of discontinuities is also often permissible.

- The condition number of integration is bounded by the distance $b - a$:
  
  
  Suppose the input function is perturbed $\hat{f} = f + \delta f$, then

  \[
  \delta I = \left| \mathcal{I}(\hat{f}) - \mathcal{I}(f) \right| \\
  \leq \left| \mathcal{I}(\delta f) \right| \\
  \leq (b - a) \| \delta f \|_\infty,
  \]

  where $\| f \|_\infty = \max_{x \in [a,b]} |f(x)|$.

  Note that this result does not depend on the magnitude of $f$ or its derivatives, which means integration is generally very well-conditioned, which makes sense since integration corresponds to averaging.
Quadrature Rules

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

$$\mathcal{I}(f) \approx Q_n(f) = \sum_{i=1}^{n} w_i f(x_i)$$

- $\{x_i\}_{i=1}^{n}$ are quadrature nodes or abscissas, $\{w_i\}_{i=1}^{n}$ are quadrature weights.
- Quadrature rule is closed if $x_1 = a, x_n = b$ and open otherwise.
- Rule is progressive if nodes of $Q_n$ are a subset of those of $Q_{n+1}$.

- For a fixed set of $n$ nodes, polynomial interpolation followed by integration give $(n - 1)$-degree quadrature rule:
  - Accuracy depends on interpolant, is exact for all $(n - 1)$-degree polynomials.
  - Can obtain weights by expressing the unique $(n - 1)$-degree polynomial interpolant in the Lagrange basis $p(x) = \sum_{i=1}^{n} \phi_i(x) f(x_i)$, so that

$$Q_n(f) = \mathcal{I}(p) = \sum_{i=1}^{n} \mathcal{I}(\phi_i) f(x_i).$$
Determining Weights in a General Basis

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

$$\mathcal{I}(f) \approx Q_n(f) = \langle w, y \rangle,$$

where $y_i = f(x_i)$

$Q_n$ is the integral of the polynomial interpolant $p$ of $(x_1, y_1), \ldots, (x_n, y_n)$.

- **Method of undetermined coefficients** obtains $y$ from *moment equations* based on Vandermonde system:

$$\mathcal{I}(p) = \mathcal{I}((\{\phi_i(x)\}_{i=1}^n, V(x, \{\phi_i\}_{i=1}^n)^{-1} y)) = \langle V(x, \{\phi_i\}_{i=1}^n)^{-T} \{\mathcal{I}(\phi_i(x))\}_{i=1}^n, y \rangle$$

- *Interpolant coefficients*

Thus to obtain $w$, we need to solve the linear system,

$$V(x, \{\phi_i\}_{i=1}^n)^T w = \left[ \int_a^b \phi_1(x) dx \quad \cdots \quad \int_a^b \phi_n(x) dx \right]^T,$$

- Note that the weights $w$ are *independent* of the function values $y$. 


Newton-Cotes Quadrature

- **Newton-Cotes** quadrature rules are defined by equispaced nodes on \([a, b]\):
  - **open**: \(x_i = a + i(b - a)/(n + 1)\), **closed**: \(x_i = a + (i - 1)(b - a)/(n - 1)\).

- The **midpoint rule** is the \(n = 1\) open Newton-Cotes rule:
  \[
  M(f) = (b - a)f\left(\frac{a + b}{2}\right)
  \]

- The **trapezoid rule** is the \(n = 2\) closed Newton-Cotes rule:
  \[
  T(f) = \frac{(b - a)}{2}(f(a) + f(b))
  \]

- **Simpson’s rule** is the \(n = 3\) closed Newton-Cotes rule:
  \[
  S(f) = \frac{b - a}{6} \left( f(a) + 4f\left(\frac{a + b}{2}\right) + f(b) \right)
  \]
Consider the Taylor expansion of $f$ about the midpoint of the integration interval $m = (a + b)/2$:

$$f(x) = f(m) + f'(m)(x - m) + \frac{f''(m)}{2}(x - m)^2 + \ldots$$

Integrating the Taylor approximation of $f$, we note that the odd terms drop:

$$\mathcal{I}(f) = f(m)(b - a) + \frac{f''(m)}{24}(b - a)^3 + O((b - a)^5)$$

Consequently, the midpoint rule is third-order accurate (first degree).
Error Estimation

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

\[ f(m) = \frac{1}{2} \left( f(a) - f'(m)(a - m) - \frac{f''(m)}{2}(a - m)^2 + \ldots ight. \]
\[ + f(b) - f'(m)(b - m) - \frac{f''(m)}{2}(b - m)^2 + \ldots \right) \]

\[ \mathcal{I}(f) = T(f) - \frac{f''(m)}{12}(b - a)^3 - O((b - a)^5) \]
\[ - 2E(f) \]

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

\[ T(f) - M(f) \approx 3E(f). \]

This approximation rapidly becomes accurate as \( b - a \) decreases.
We can bound the error for an arbitrary polynomial quadrature rule by

\[ |\mathcal{I}(f) - Q_n(f)| = |\mathcal{I}(f - p)| \leq (b - a)\|f - p\|_\infty \leq \frac{b - a}{4n} h^n \|f^{(n)}\|_\infty \]

where \( h = \max_i (x_{i+1} - x_i) \).
We can ascertain stability of quadrature rules, by considering the amplification of a perturbation $\hat{f} = f + \delta f$:

$$|Q_n(\hat{f}) - Q_n(f)| = |Q_n(\delta f)|$$
$$= \sum_{i=1}^{n} w_i \delta f(x_i)$$
$$\leq ||w||_1 ||\delta f||_\infty.$$

Note that we always have $\sum_i w_i = b - a$, since the quadrature rule must be correct for a constant function. So if $w$ is positive $||w||_1 = b - a$, the quadrature rule is stable, i.e. it matches the conditioning of the problem.

Newton-Cotes quadrature rules have at least one negative weight for any $n \geq 11$: More generally, $||w||_1 \to \infty$ as $n \to \infty$ for fixed $b - a$. This means that the Newton-Cotes rules can be ill-conditioned.
To obtain a more stable quadrature rule, we need to ensure the integrated interpolant is well-behaved as $n$ increases:

- Chebyshev quadrature nodes ensure that interpolant polynomial has bounded coefficients so long as $f$ is bounded, since the Vandermonde system defining its coefficients is well-conditioned.
- Formally, it can be shown that $w_i > 0$ for the Chebyshev-node (Clenshaw-Curtis) quadrature.
- The weights for Clenshaw-Curtis quadrature rules can be obtained by solutions to Vandermonde systems on $[-1, 1]$ with Chebyshev-spaced nodes, then translating to a desired integration interval.
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:
  - Choice of nodes gives additional \( n \) parameters for total \( 2n \) degrees of freedom.
  - Permits exact integration of degree-(\( 2n - 1 \)) polynomials and corresponding general 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 \( w \):
  
  Given any complete basis, we seek to solve the nonlinear equations for \( x, w \),

  \[
  V(x, \{\phi_i\}_{i=1}^{2n+1})^T w = y, \quad \text{where} \quad y_i = I(\phi_i).
  \]

- These nonlinear equations generally have a unique solution \((x^*, w^*)\).
- For fixed \( x \), we have an overdetermined system of linear equations for \( w \).
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 \([0, 1]\).
  - We can transform a given integral using variable substitution \( t = \frac{x-a}{b-a} \),
    \[
    \mathcal{I}(f) = \int_{a}^{b} f(x) \, dx = (b-a) \int_{0}^{1} g(t) \, dt \quad \text{where} \quad g(t) = f(t(b-a) + a)).
    \]
  - For quadrature rules defined on \([-1, 1]\), we can transform via the substitution \( t = 2 \frac{x-a}{b-a} - 1 \),
    \[
    \mathcal{I}(f) = \int_{a}^{b} f(x) \, dx = \frac{b-a}{2} \int_{-1}^{1} g(t) \, dt \quad \text{where} \quad g(t) = f((t+1)(b-a)/2 + a)).
    \]
- Gaussian quadrature rules are accurate and stable but not progressive (nodes cannot be reused to obtain higher-degree approximation):
  - maximal degree is obtained
  - weights are always positive (perfect conditioning)
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 \(G_n\):
  - Gaussian quadrature rule \(G_{2n+1}\) would use same number of points and have degree \(4n + 1\).
  - Kronod rule points are optimal chosen to reuse all points of \(G_n\), so \(n + 1\) rather than \(2n + 1\) new evaluations are necessary.

- **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: *Gauss-Radau uses one of two end-points as a node, while Gauss-Lobatto quadrature uses both.*
Composite and Adaptive Quadrature

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

  For example, we can derive simple composite Newton-Cotes rules by partitioning the domain into sub-intervals $[x_i, x_{i+1}]$:

  - **composite midpoint rule**

    $$\mathcal{I}(f) = \sum_{i=1}^{n-1} \int_{x_i}^{x_{i+1}} f(x)dx \approx \sum_{i=1}^{n-1} (x_{i+1} - x_i) f((x_{i+1} + x_i)/2)$$

  - **composite trapezoid rule**

    $$\mathcal{I}(f) = \sum_{i=1}^{n-1} \int_{x_i}^{x_{i+1}} f(x)dx \approx \sum_{i=1}^{n-1} \frac{(x_{i+1} - x_i)}{2} (f(x_{i+1}) + f(x_i))$$

- Composite quadrature can be done with adaptive refinement:

  *Introduce new nodes where error estimate is large. Error estimate can be obtained by e.g. comparing trapezoid and midpoint rules, but can be completely wrong if function is insufficiently smooth.*
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. *Composite multidimensional rules are also possible by partitioning the domain into chunks.*

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

\[
\int_{\Omega} f(\mathbf{x}) \, d\mathbf{x} = E[Y], \quad Y = \frac{|\Omega|}{N} \sum_{i=1}^{N} Y_i, \quad Y_i = f(\mathbf{x}_i), \quad \mathbf{x}_i \text{ chosen randomly from } \Omega.
\]

- Convergence rate is independent of function (effective polynomial degree approximation) or dimension of integration domain.

- Instead, it depends on number of samples \((N)\), with error scaling as \(O(1/\sqrt{N})\).
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

\[ \int_{a}^{b} K(s, t)u(t)\,dt = f(s), \quad \text{where } K \text{ and } f \text{ are known.} \]

- Useful for recovering signal \( u \) given response function with kernel \( K \) and measurements of \( f \).

- Also arise from solve equations arising from Green’s function methods for PDEs.

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

\[ \sum_{j=1}^{n} w_j K(s, t_j)u(t_j) = f(s). \]

Discrete sample of \( f \) on \( s_1, \ldots, s_n \) yields a linear system of equations,

\[ \sum_{j=1}^{n} w_j K(s_i, t_j)u(t_j) = f(s_i). \]
Numerical Differentiation

- Automatic (symbolic) differentiation is a surprisingly viable option:
  - Any computer program is differentiable, since it is an assembly of basic arithmetic operations.
  - Existing software packages can automatically differentiate whole programs.

- Numerical differentiation can be done by interpolation or finite differencing:
  - Given polynomial interpolant, its derivative is easy to obtain by differentiating the basis in which it is expressed,
    \[
    f'(x) \approx p'(x) = \left[ \phi'_1(x) \cdots \phi'_n(x) \right]^T V(t, \{\phi_i\}_{i=1}^n)^{-1} y, \text{ where } y_i = f(t_i).
    \]
  - Obtaining the values of the derivative at the interpolation nodes, can be done via
    \[
    \underbrace{V(t, \{\phi'_i\}_{i=1}^n) V(t, \{\phi_i\}_{i=1}^n)^{-1}}_{\text{Differentiation matrix}} y, \text{ where } y_i = f(t_i).
    \]
  - Finite-differencing formulas effectively use linear interpolant.
Accuracy of Finite Differences

- **Forward and backward differencing** provide first-order accuracy:
  These can be derived, respectively from forward and backward Taylor expansions of $f$ about $x$,

  \[
  f(x + h) = f(x) + f'(x)h + f''(x)h^2/2 + \ldots \\
  f(x - h) = f(x) - f'(x)h + f''(x)h^2/2 - \ldots
  \]

  For forward differencing, we obtain an approximation from the first equation,

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

- **Centered differencing** provides second-order accuracy. Subtracting the backward Taylor expansion from the forward, we obtain centered differencing,

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

  Second order accuracy is due to cancellation of odd terms like $f''(x)h/2$. 

Demo: Finite Differences vs Noise

Demo: Floating point vs Finite Differences
Extrapolation Techniques

- Given a series of approximate solutions produced by an iterative procedure, a more accurate approximation may be obtained by **extrapolating** this series. *For example, as we lower the step size \( h \) in a finite-difference formula, we can try to extrapolate the series to \( h = 0 \), if we know that*

\[
F(h) = a_0 + a_1 h^p + O(h^r) \text{ as } h \to 0 \text{ and seek to determine } F(0) = a_0,
\]

for example in centered differences \( p = 2 \) and \( r = 4 \).

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

  *Seek to eliminate \( a_1 h^p \) term in \( F(h) \), \( F(h/2) \) to improve approximation of \( a_0 \),*

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

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

    \[
    a_0 = F(h) - \frac{F(h) - F(h/2)}{1 - 1/2^p} + O(h^r).
    \]
Given a series of $k$ approximations, Romberg integration applies $(k - 1)$-levels of Richardson extrapolation. Can apply Richardson extrapolation to each of $k - 1$ pairs of consecutive nodes, then proceed recursively on the $k - 1$ resulting approximations.

Extrapolation can be used within an iterative procedure at each step: For example, Steffensen’s method for finding roots of nonlinear equations,

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

derived from Aitken’s delta-squared extrapolation process:

- achieves quadratic convergence,
- requires no derivative,
- competes with the Secant method (quadratic versus superlinear convergence, but an extra function evaluation necessary).