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).
Integrability and Sensitivity

- Function $f$ is integrable if continuous and bounded, in practice a finite number of discontinuities is also ok:
  
  Seek to compute $I(f) = \int_{a}^{b} f(x) \, dx$, define $\|f\|_\infty = \max_{x \in [a,b]} |f(x)|$

- The condition number of integration is bounded by the distance $b - a$:
  
  Suppose the input function is perturbed $\hat{f} = f + \delta f$, then
  \[
  \delta I = |I(\hat{f}) - I(f)| 
  \leq |I(\delta f)| 
  \leq (b - a)\|\delta f\|_\infty
  \]

  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

- To approximate the integral $I(f)$, compute a weighted sum of points:

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

- $\{x_i\}_{i=1}^{n}$ 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, unique quadrature weights give exact $(n-1)$-degree quadrature rule:

The rule is exact for all $(n-1)$-degree polynomials. Express the unique $(n-1)$-degree polynomial interpolant in the Lagrange basis $p(x) = \sum_{i=1}^{n} \phi_i(x) f(x_i)$. The quadrature rule is defined by

$$Q_n(f) = I(p(x)) = \sum_{i=1}^{n} \left( \sum_{i=1}^{n} w_i f(x_i) \right)$$
Quadrature Rules and Error

- Quadrature weights can be alternatively determined for a rule by solving the moment equations:

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

- We can approximate the error bound for a polynomial quadrature rule by

\[
|I(f) - Q_n(f)| = |I(f - p_{n-1}(x))| \\
\leq (b - a)\|f - p_{n-1}\|_{\infty} \\
\leq \frac{b - a}{4n} h^n \|f^{(n)}\|_{\infty}
\]

where \( h = \max_i(x_{i+1} - x_i) \)
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:

  $$S(f) = \frac{b - a}{6} \left( f(a) + 4f \left( \frac{a + b}{2} \right) + f(b) \right)$$
Error in Newton-Cotes Quadrature

- 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,

\[
I(f) = f(m)(b - a) + f''(m) \left( \frac{b - a}{24} \right)^3 + O((b - a)^5)
\]

- The midpoint rule is third-order accurate (first degree).
- The trapezoid rule is also first degree, despite using higher-degree polynomial interpolant approximation.
- Error can be conveniently approximated by difference of two rules,

\[
T(f) - M(f) \approx 3E(f).
\]
Conditioning of Newton-Cotes Quadrature

- 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 Chebyshev-node (Clenshaw-Curtis) quadrature.*
A quadrature rule provides $x$ and $w$ so as to approximate

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

where $y_i = f(x_i)$

$Q_n$ integrates the $(n - 1)$-degree polynomial interpolant through $f$, We note that $y$ can be obtained from the Vandermonde system,

$$\langle w, y \rangle = Q_n(f) = I(p_{n-1}) = \left[ \int_a^b \phi_1(x)dx \cdots \int_a^b \phi_n(x)dx \right] V(x, \{\phi_i\}_{i=1}^n)^{-1} y.$$

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

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

which is independent of $y$. 

Gaussian Quadrature

- So far, we have only considered quadrature rules based on a fixed set of nodes, but we can also choose a set of nodes to improve accuracy:
  
  *Choice of nodes gives additional $n$ parameters for a total of $2n$ degrees of freedom, permitting representation of polynomials of degree $2n - 1$.

- 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,

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

  *For fixed $x$, we have an overdetermined system of linear equations for $w$, but these nonlinear equations generally have a unique solution $(x^*, w^*)$. 
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 the integral as follows,

$$I(f) = \int_{a}^{b} f(x) \, dx = \int_{0}^{1} g(t) \, dt \text{ where } f(x) = g \left( \frac{x + b - a}{b - a} \right).$$

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 quadrature $K_{2n+1}$ that is progressive w.r.t. Gaussian quadrature rule $G_n$

  - $(2n + 1)$-point Kronod rule is degree $3n + 1$, Gaussian quadrature rule would be of 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**
    
    $$I(f) = \sum_{i=1}^{n-1} \int_{x_i}^{x_{i+1}} f(x) dx \approx \sum_{i=1}^{n-1} (x_{i+1} - x_i) f((x_{i+1} + x_i)/2)$$

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

- 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 most often done by *Monte Carlo* integration:

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

*convergence rate is independent of dimension of } x (n) \text{ only 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

$$\int_a^b K(s, t)u(t)dt = f(s), \text{ where } K \text{ and } f \text{ are known.}$$

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

$$\sum_{j=1}^{n} w_jK(s, t_j)u(t_j) = f(s).$$

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

$$\sum_{j=1}^{n} w_jK(s_i, t_j)u(t_j) = f(s_i).$$

- Integral equations are used to
  - recover signal $u$ given response function with kernel $K$ and measurements of $f$,
  - solve equations arising from Green’s function methods for PDEs.
Integral equations based on response functions tend to be ill-conditioned, which is resolved using

- truncated singular value decomposition of $A$, where $a_{ij} = w_j K(s_i, t_j)$
- replacing the linear system with a regularized linear least squares problem,
- expressing the solution using a basis

Let $u(t) \approx \sum_{j=1}^{n} c_j \phi_j(t)$ and derive equations for the coefficients.
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.

\[
f'(x) \approx p_{n-1}'(x) = \left[ \phi'_1(x) \cdots \phi'_n(x) \right]^T \mathbf{V}(t, \{\phi_i\}_{i=1}^n)^{-1} \mathbf{y}, \text{ where } y_i = f(t_i).
\]

- Finite-differencing formulas effectively use linear interpolant.
Accuracy of Finite Differences

- Forward and backward differences provide first-order accuracy:

  These can be derived using two forms of the Taylor expansion 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: Using a sum of the two Taylor expansions, or equivalently a difference between the forward- and backward-differencing formulas, 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 \( f''(x)h/2 \).
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 achieves quadratic convergence using Aitken's delta-squared extrapolation process. The method requires no derivative and competes with the Secant method (quadratic versus superlinear convergence, but an extra function evaluation necessary).