CS 450: Numerical Analysis

Lecture 21
Chapter 7 Numerical Integration and Differentiation
Gaussian Quadrature, Integral Equations, and Numerical Differentiation

Edgar Solomonik

Department of Computer Science
University of Illinois at Urbana-Champaign

April 6, 2018
Quadrature Rules

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

$$I(f) \approx Q_n(f) = \langle w, y \rangle, \quad \text{where} \quad y_i = f(x_i)$$

$$I(f) = \int_a^b f(x) \, dx$$

$Q_n(f) = I(p_{n-1}) = \left[ I(e_1(x)) \cdots I(e_n(x)) \right] V^{-1}(x, \Psi e_i x_i = 1)$

$p_{n-1}$ degree interpolant of $f$

$$\int_{p_n(x)} \, dx = \sum_{i=1}^n \int_{c_i(x)} \, dx$$

$$I(p_{n-1}) = s^T V^{-1} y = \langle w, y \rangle$$

$s^T V^{-1} = w^T$

$s^T = w^T V \Rightarrow Vw = s$
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:

  Estimate \( \int f \) by optimal choice of nodes and weights. Gaussian quadrature rules use \( n \) points to obtain 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 \( \omega \):

  \[
  V(\mathbf{x}, \mathbf{e}_{2n-1}) = \begin{bmatrix}
  e_1(x_1) & \cdots & e_{2n-1}(x_1) \\
  \vdots & \ddots & \vdots \\
  e_1(x_n) & \cdots & e_{2n-1}(x_n)
  \end{bmatrix}
  \]

  \[
  V(\mathbf{x}, \mathbf{e}_{2n-1})^T s = 0 \quad \text{for \( s \) from \( x \) is over determined.}
  \]
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 \([0, 1]\).

\[
J(f) = \int_a^b f(x) \, dx = \frac{b-a}{2} \int_0^1 g(t) \, dt
\]

\[
f(x) = g\left(\frac{x + a - b}{b - a}\right)
\]

- Gaussian quadrature rules are accurate and stable but not progressive (nodes cannot be reused to obtain higher-degree approximation).

\[
\text{minimal interpolation degree} \quad \text{so } C_n \text{ and } C_m \text{ for } m \neq n \text{ have no nodes in common}
\]
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 \(G_n\).

  \[ K_{2n+1} \text{ to have the } n \text{ nodes of } G_n \]

  \[ K_{2n+1} \text{ while maximizing degree, which is } 3n+1. \]

  \[ G_{2n+1} \text{ has degree } 2(2n+1) - 1 = 4n+1 \]

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

  - Gauss-Radau include 1 end-point
  - Gauss-Lobatto include a and b
Composite and Adaptive Quadrature

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

\[
I(f) = \sum_{i=1}^{n} (-1)^{i+1} f(x_i) \frac{x_{i+1} - x_i}{2} \]

- Composite quadrature can be done with adaptive refinement:

We can estimate error, e.g., compare midpoint and trapezoidal.

Adaptive mesh 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.

\[
\iint_{a \leq x \leq b, c \leq y \leq d} f(x, y) \, dx \, dy, \quad \sum_{n=1}^{N} f(x_n, y_n) \, dx \, dy
\]

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

\[
\iint_{x \in [a, b]} f(x) \, dx \approx \frac{1}{N} \left| \bigcup_{i=1}^{N} \mathcal{Y}_i \right|, \quad \mathcal{Y}_i = f(x_i) \text{ for } i \in [1, N]
\]

\[\text{error } \leq O\left(\frac{1}{\sqrt{N}}\right)\]
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

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

- 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.
Challenges in Solving Integral Equations

- 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,

\[
Aw = y, \quad A = \begin{bmatrix} \vdots & \vdots \\ \vdots & \vdots \\ \end{bmatrix}, \quad \text{ignore small singular values of } A
\]

\[
\min_{w} \| Aw - y \| + \lambda \| w \| \Rightarrow \min_{w} \left\| \begin{bmatrix} A \\ \delta \mathbf{1} \end{bmatrix} w - \mathbf{0} \right\|
\]

\[
u(t) = \sum_{i=1}^{\infty} c_i \phi_i(t)
\]
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.

\[ f'(s) \sim [e'_1(s), \ldots, e'_n(s)] \sqrt{\left(\sum_{i=1}^{n} e_{1i}^2 + \sum_{i=1}^{n} e_{2i}^2\right)} y \]

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

- Forward and backward differences provide first-order accuracy:
  \[
  f(x + h) = f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \ldots
  \]
  \[
  f(x - h) = f(x) - hf'(x) + \frac{h^2}{2} f''(x) - \ldots
  \]

- Centered differencing provides second-order accuracy:
  \[
  f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2)
  \]
  Second-order convergence
Extrapolation Techniques

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

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

\[
\frac{f(x) - f(a) - h}{h} \quad \text{for forward diff.}
\]

\[
f''(x)/2 \quad \text{for forward diff.}
\]

given \( F(h) \) and \( F(h/2) \), would like \( F(0) \)

\[
F(h) = a_0 + a_1 h^p + O(h^q) \quad \text{where } q > p
\]
\[
\int_{f(h/2)} = a + e \cdot e^{h/2} + O(h)
\]

\[
F(h) - F(h/2) = \frac{a}{2} + O(h)
\]

\[
\frac{\partial F(h)}{\partial h} = \frac{1}{2} \cdot \frac{a}{2} + O(h)
\]

\[
\frac{\partial F(h)}{\partial h} = \frac{1}{2} \cdot \frac{a}{2} + O(h)
\]
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:

For solving nonlinear systems

- An $\Delta^2$-process $\rightarrow$ Steffensen's method
- quadratic convergence without derivatives
- alternative to Secant method