UNIVERSITY AT ALBANY
STATE UNIVERSITY OF NEW YORK

CSI 401 (Fall 2025)

Numerical Methods

Lecture 16: Numerical Integration

Chong Liu

Department of Computer Science

Nov 17, 2025

Recap: Problem setup of Interpolation

- For given data
  - $(t_1, y_1), (t_2, y_2), \dots, (t_m, y_m)$ with $t_1 < t_2 < \cdots < t_m$
- determine function $f: R \to R$ such that
  - $f(t_i) = y_i, \forall i = 1, \dots, m$
  - Exactly crossing all data points!

- f is **interpolating function**, or **interpolant**, for given data.
  - f could be function of more than one variable, but let's focus on the 1-dimensional case first.

Recap: Basis Functions

- Family of functions for interpolating:
  - Set of basis functions $\phi_1(t), \ldots, \phi_n(t)$
- Interpolating function f is chosen as linear combination of them

$$f(t) = \sum_{j=1}^{n} x_j \phi_j(t)$$

- Requiring f to interpolate data (t_i, y_i) means

$$f(t_i) = \sum_{j=1}^{n} x_j \phi_j(t_i) = y_i, \quad i = 1, \ldots, m$$

  - Discussion: What is this system?
    - A system of linear equations $Ax = y$ for n-vector x of parameters x_j, where entries of $m \times n$ matrix A are given by $a_{ij} = \phi_j(t_i)$.

Recap: Basic polynomial interpolation

- Basis functions

$$\phi_j(t) = t^{j-1}, \quad j = 1, \ldots, n$$

- give interpolating polynomial of form

$$p_{n-1}(t) = x_1 + x_2 t + \cdots + x_n t^{n-1}$$

- with coefficients x given by $n \times n$ linear system

$$\boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} 1 & t_1 & \cdots & t_1^{n-1} \\ 1 & t_2 & \cdots & t_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_n & \cdots & t_n^{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \boldsymbol{y}$$

Recap: Lagrange interpolation

- Assuming common factor $(t_i - t_j)$ in $\ell(t_j)/(t_i - t_j)$ is canceled to avoid division by zero when evaluating $\ell_j(t_i)$, then

$$\ell_j(t_i) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}, \qquad i, j = 1, \ldots, n$$

  - Matrix of linear system $Ax = y$ is identity matrix I
  - Coefficients x for Lagrange basis functions are just data values y

- Polynomial of degree $n - 1$ interpolating data points $(t_i, y_i), i = 1, \ldots, n$ is given by

$$p_{n-1}(t) = \sum_{j=1}^{n} y_j \ell_j(t) = \sum_{j=1}^{n} y_j \ell(t) \frac{w_j}{t - t_j} = \ell(t) \sum_{j=1}^{n} y_j \frac{w_j}{t - t_j}$$

Recap: Newton interpolation

- For given set of data points $(t_i, y_i), i = 1, \dots, n$, Newton basis functions are defined by

$$\pi_j(t) = \prod_{k=1}^{j-1}(t - t_k), \quad j = 1, \dots, n$$

- Newton interpolating polynomial has form

$$\begin{aligned} p_{n-1}(t) \quad &= \quad x_1 + x_2(t - t_1) + x_3(t - t_1)(t - t_2) + \\ &\quad\ \cdots + x_n(t - t_1)(t - t_2)\cdots(t - t_{n-1}) \end{aligned}$$

  - For $i < j, \pi_j(t_i) = 0$, so basis matrix A is lower triangular, where $a_{ij} = \pi_j(t_i)$.

Recap: Piecewise polynomial interpolation

- Motivation:
  - Fitting single polynomial to large number of data points is likely to yield unsatisfactory behavior in interpolant
- Main advantage:
  - Large number of data points can be fit with low-degree polynomials
- How:
  - Given data points (t_i, y_i), different function is used in each subinterval $[t_i, t_{i+1}]$
    - t_i is called knot or breakpoint, at which interpolant changes from one function to another

Recap: Spline interpolation

- A spline is a smooth piecewise polynomial function.
  - Two popular model:
    - Quadratic spline, Cubic spline
- Quadratic spline interpolation
  - each segment is a **second-degree polynomial** function.
  - Formally, we have data points $(t_i, y_i), i = 1, \dots, n$
  - For each interval $[t_i, t_{i+1}]$, we define a quadratic polynomial
    - $f_i(t) = a_i + b_i(\mathrm{t} - t_i) + c_i(t - t_i)^2.$
    - There are $n - 1$ such polynomials (one per interval).
  - Discussion: how many coefficients need to be determined? How many equations do we need?
    - $3(n - 1)$

Recap: Summary of interpolation

- Interpolating function fits given data points exactly, which is not appropriate if data are noisy
- Interpolating function given by linear combination of basis functions, whose coefficients are to be determined
- Existence and uniqueness of interpolant depend on whether number of parameters to be determined matches number of data points to be fit
- Piecewise polynomial (e.g., spline) interpolation can fit large number of data points with low-degree polynomials
- Cubic spline interpolation is excellent choice when smoothness is important

Agenda

- Problem setup of numerical integration
- Methods of numerical integration:
  - Method of Undetermined Coefficients
  - Newton-Cotes Quadrature
  - Composite Quadrature

Problem setup of numerical integration

- For f: $R \to R$, definite integral over interval $[a, b]$

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

  - is defined by limit of Riemann sums

$$R_n = \sum_{i=1}^{n} (x_{i+1} - x_i)\, f(\xi_i)$$

  - Riemann integral exists provided integrand f is bounded and continuous almost everywhere
- Key question today: How can we use computers to calculate the integration by querying f only?
- Discussion: What's your idea?

Numerical Quadrature

- Quadrature rule is weighted sum of finite number of sample values of integrand function

- To obtain desired level of accuracy at low cost,

  - How should sample points be chosen?

  - How should their contributions be weighted?

Quadrature Rules

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

- An n-point quadrature rule has form
  - Points x_i are called nodes
  - Multipliers w_i are called weights

- Quadrature rules are based on polynomial interpolation
  - Integrand function f is sampled at finite set of points
  - Integral of interpolant is taken as estimate for integral of original function

- In practice, interpolating polynomial is not determined explicitly but used to determine weights corresponding to nodes

Method of Undetermined Coefficients

- To derive n-point rule on interval $[a, b]$, take nodes $x_1, \ldots, x_n$ as given and consider weights $w_1, \ldots, w_n$ as coefficients to be determined

- Example:
  - Derive 3-point rule

$$Q_3(f) = w_1 f(x_1) + w_2 f(x_2) + w_3 f(x_3)$$

    - at interval [a, b] using monomial basis $1, x, x^2$
  - Take $x_1 = a, x_2 = \frac{a+b}{2}, x_3 = b$ as nodes

Method of Undetermined Coefficients

- Resulting system of equations

$$
\begin{aligned}
w_1 \cdot 1 + w_2 \cdot 1 + w_3 \cdot 1 &= \int_a^b 1\,dx = x|_a^b = b - a \\
w_1 \cdot a + w_2 \cdot (a+b)/2 + w_3 \cdot b &= \int_a^b x\,dx = (x^2/2)|_a^b = (b^2 - a^2)/2 \\
w_1 \cdot a^2 + w_2 \cdot ((a+b)/2)^2 + w_3 \cdot b^2 &= \int_a^b x^2\,dx = (x^3/3)|_a^b = (b^3 - a^3)/3
\end{aligned}
$$

  - In matrix form:

$$
\begin{bmatrix} 1 & 1 & 1 \\ a & (a+b)/2 & b \\ a^2 & ((a+b)/2)^2 & b^2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} b-a \\ (b^2 - a^2)/2 \\ (b^3 - a^3)/3 \end{bmatrix}
$$

- Solving system by Gaussian elimination, we obtain weights

$$
w_1 = \frac{b-a}{6}, \qquad w_2 = \frac{2(b-a)}{3}, \qquad w_3 = \frac{b-a}{6}
$$

  - Also knows as the Simpson rule

Method of Undetermined Coefficients

- More generally, if we have n points, we solve the following systems to get weights

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} b - a \\ (b^2 - a^2)/2 \\ \vdots \\ (b^n - a^n)/n \end{bmatrix}$$

Newton-Cotes Quadrature

- Midpoint rule

$$M(f) = (b-a)\, f\left(\frac{a+b}{2}\right)$$

- Trapezoid rule

$$T(f) = \frac{b-a}{2}\,(f(a) + f(b))$$

- Simpson's rule

$$S(f) = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)$$

In-class exercise

- Use three different rules of Newton-Cotes Quadrature to approximate the integral
- $I(f) = \int_0^1 \exp(-x^2)\, dx$
- Solution:

$$
\begin{aligned}
M(f) &= (1-0)\exp(-1/4) \approx 0.778801 \\
T(f) &= (1/2)[\exp(0) + \exp(-1)] \approx 0.683940 \\
S(f) &= (1/6)[\exp(0) + 4\exp(-1/4) + \exp(-1)] \approx 0.747180
\end{aligned}
$$

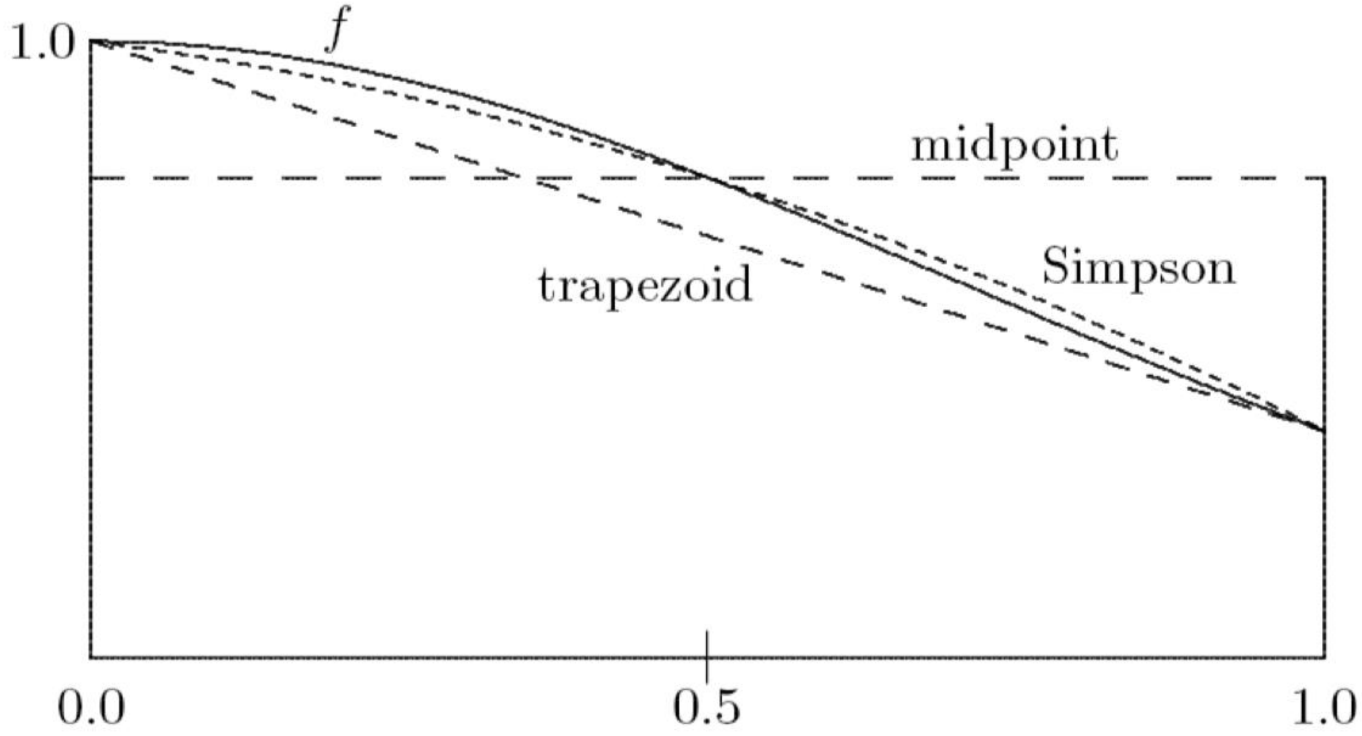
- True value: $I(f) = \int_0^1 \exp(-x^2)\, dx \approx 0.746824$

Illustration of Newton-Cotes Quadrature

$$M(f) = (b-a)\, f\left(\frac{a+b}{2}\right)$$

$$T(f) = \frac{b-a}{2}\,(f(a) + f(b))$$

$$S(f) = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)$$



Composite Quadrature

- Alternative to using more nodes and higher degree rule is to subdivide original interval into subintervals, then apply simple quadrature rule in each subinterval

- Summing partial results then yields approximation to overall integral

- This approach is equivalent to using piecewise interpolation to derive composite quadrature rule

- Approximate integral converges to exact integral as number of subintervals goes to infinity

Composite Quadrature

- Subdivide interval $[a, b]$ into k subintervals
  - Length $h = (b - a)/k$, for $x_j = a + jh, j = 0, \ldots, k$
- Composite midpoint rule

$$M_k(f) = \sum_{j=1}^{k} (x_j - x_{j-1})\, f\left(\frac{x_{j-1} + x_j}{2}\right) = h \sum_{j=1}^{k} f\left(\frac{x_{j-1} + x_j}{2}\right)$$

- Composite trapezoid rule

$$\begin{aligned} T_k(f) &= \sum_{j=1}^{k} \frac{(x_j - x_{j-1})}{2} \left(f(x_{j-1}) + f(x_j)\right) \\ &= h\left(\tfrac{1}{2} f(a) + f(x_1) + \cdots + f(x_{k-1}) + \tfrac{1}{2} f(b)\right) \end{aligned}$$

In-class exercise

- Use composite midpoint rule and composite trapezoid rule to approximate the integral
  - $I = \int_0^2 (1 + x^3)\, dx$
  - With 4 subintervals
- Solutions:

$$
\begin{aligned}
f(0.25) &= 1 + 0.015625 = 1.015625,\\
f(0.75) &= 1 + 0.421875 = 1.421875,\\
f(1.25) &= 1 + 1.953125 = 2.953125,\\
f(1.75) &= 1 + 5.359375 = 6.359375.
\end{aligned}
$$

$$
\begin{aligned}
M_4 &= h \sum_{i=1}^{4} f(m_i) = 0.5(1.015625 + 1.421875 + 2.953125 + 6.359375)\\
&= 5.875.
\end{aligned}
$$

$$
\begin{aligned}
f(0) &= 1,\\
f(0.5) &= 1.125,\\
f(1) &= 2,\\
f(1.5) &= 4.375,\\
f(2) &= 9.
\end{aligned}
$$

$$
\begin{aligned}
T_4 &= \frac{h}{2}\left[f(0) + 2(f(0.5) + f(1) + f(1.5)) + f(2)\right]\\
&= 6.25.
\end{aligned}
$$

Summary

- Integral is approximated by weighted sum of sample values of integrand function

- Nodes and weights chosen to achieve required accuracy at least cost (fewest evaluations of integrand)

- Quadrature rules derived by integrating polynomial interpolant
  - Newton-Cotes rules use equally spaced nodes and choose weights to maximize polynomial degree

- Composite Quadrature divides original interval into subintervals
  - Works using piecewise interpolation