Math 104A - Intro to Numerical Analysis

Numerical Differentiation and Integration

Jea-Hyun Park

University of California Santa Barbara

Fall 2022

Numerical Differentiation and Integration

Numerical Differentiation and Integration

Intro

Numerical Differentiation

Problem of interest

Given the access to function values can we suggest something close to $f'(x)$?

Numerical Differentiation and Integration

Numerical differentiation

Difference quotients

The definition of the derivative, $f'(x) = \lim_{h\to 0} \frac{f(x+h)-f(x)}{h}$, motivates the following approximation.

Forward/Backward difference quotient

For a small $h > 0$,

$$f'(x) \approx \frac{f(x+h)-f(x)}{h} \qquad \text{(forward difference quotient)}$$

$$f'(x) \approx \frac{f(x)-f(x-h)}{h} \qquad \text{(backward difference quotient)}$$

- Backward difference quotient is obtained by setting $h \leftarrow -h$.

Difference quotients

Theorem (Order of forward/backward difference quotient)

If f is smooth enough,

$$\frac{f(x+h)-f(x)}{h} = f'(x) + \frac{f''(\xi)}{2}h$$

$$\frac{f(x)-f(x-h)}{h} = f'(x) - \frac{f''(\hat{\xi})}{2}h,$$

where $\xi \in (x, x+h)$, and $\hat{\xi} \in (x-h, x)$.

Proof.

Board work. □

- This tells more than the previous one:

$$\frac{f(x+h)-f(x)}{h} - f'(x) = \frac{f''(\xi)}{2}h$$

- "How fast" does not tell you "how close." For "how close," you also need $f''(\xi)$, which is usually not available.
- Usually, a faster method (i.e., higher order) is considered better if other factors are the same.

Centered difference quotient

Theorem (Centered difference quotient)

If f is smooth enough and $h > 0$ is small enough,

$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + \frac{f'''(\xi)}{6} h^2$$

where $\xi \in (x-h, x+h)$.

Numerical differentiation using interpolation

Given the nodes $x_0, x_1, \cdots, x_n$, we know from Lagrange interpolation theorem

$$f(x) = \sum_{i=0}^{n} f(x_i)\, \ell_i(x) + \frac{1}{(n+1)!} f^{(n+1)}(\xi_x)\, w(x),$$

where $w(x) = \prod_{i=0}^{n}(x - x_i)$ and $\ell_i = \prod_{j\neq i}^{n}(x - x_j)/(x_i - x_j)$. If we are interested in one of the nodes, say x_k, differentiating this and evaluating at x_k,

$$f'(x_k) = \sum_{i=0}^{n} f(x_i)\, \ell_i'(x_k) + \frac{1}{(n+1)!} f^{(n+1)}(\xi_{x_k}) \prod_{\substack{j=0 \\ j\neq k}}^{n} (x_k - x_j)$$

- This argument is not rigorously true though may be acceptable for a practical reason: see the next slide.

Numerical differentiation using interpolation

If $x \neq x_0, x_1, \cdots, x_n$, the previous argument doesn't work even in the practical sense.

$$f'(x) = \sum_{i=0}^{n} f(x_i)\,\ell_i'(x) + \frac{1}{(n+1)!} f^{(n+1)}(\xi_x)\, w'(x) + \frac{1}{(n+1)!} w(x) \frac{d}{dx} f^{(n+1)}(\xi_x)$$

Fortunately, there is a way to obtain a similar result but it take a bit more work. In particular, we have a good result for smooth function.

Theorem (Suli and Mayers (2003))

If $f \in C^\infty$, letting $M_{n+1} = \max_{x\in[a,b]} \left|f^{(n+1)}(x)\right|$, for all $x \in [a,b]$,

$$|f'(x) - p_n'(x)| \leq \frac{(b-a)^n M_{n+1}}{n!},$$

where $p_n(x)$ is the Lagrange interpolation at $x_0, x_1, \cdots, x_n$.

- The factor in red is problematic ξ_x may not smoothly depend on x.
- If interested, see Süli and Mayers, *An introduction to numerical analysis*, Theorem 6.5 and Corollary 6.1.

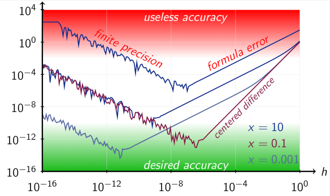
Cautions

Caveat: Rounding error may destroy everything if you pursue too much precision.

Example: (Centered difference quotient)

- $f'(x) = \frac{f(x+h)-f(x-h)}{2h} + \mathcal{O}(h^2)$ tells you the smaller h gets the more accurate the approximation is.
- Every time you store something you have rounding error. Thus, what you really compute is $\frac{f(x+h)+e_1-f(x-h)-e_2}{2h} = \frac{f(x+h)-f(x-h)}{2h} + \frac{e_1-e_2}{2h}$.
- 2nd term may amplify the error as h gets smaller.
- Or, the numerator may just be zero to the computer since they are so close: leading to $f'(x) = 0$.

- See Matlab example.



Richardson's extrapolation

If you have a numerical method in the following form, you can accelerate it using **Richardson's extrapolation**.
Example: Centered difference quotient (Board work)
Start with Taylor expansion so that

$$f'(x) = \varphi(h) - a_2 h^2 - a_4 h^4 - \cdots ,$$

where $\varphi(h) = \frac{f(x+h)-f(x-h)}{2h}$, $a_2 = \frac{f^{(3)}(x)}{3!}$ and $a_4 = \frac{f^{(5)}(x)}{5!}$ and so on.
Plug in $h \leftarrow h/2$ and do something similar to something in middle school.
Then we get (Board work)

$$f'(x) = \frac{4}{3}\varphi(h/2) - \frac{1}{3}\varphi(h) - a_4 h^4/4 - 5a_6 h^6/16 - \cdots$$

- This process can be repeated to achieve higher accuracy, say put $\psi(h) := \frac{4}{3}\varphi(h/2) - \frac{1}{3}\varphi(h)$ and cancel h^4–term.
- This process can be applied to other methods that have the same structure.
- **Subjective question**: Does this look like a magic to you? Or, not exactly? Give the reason.

Numerical Differentiation and Integration

Quadrature based on interpolation

Numerical Integration (Quadrature)

Problem of interest

Given the access to function values can we suggest something close to $\int_a^b f(x)dx$?

- As you know, functions with integrals in closed form are very limited. E.g., $e^{-x^2/2}$, $\cos(\sin(x^2))$, etc. have no definite integral in a simple formula.

Newton-Cotes formula

Definition (Newton-Cotes)

Given $a < b$, let $x_i = a + hi$ $(i = 0, 1, \cdots, n)$, where $h = (b-a)/n$, be equally spaced nodes on $[a, b]$. Then, the *Newton-Cotes* formula for the approximate integral of $\int_a^b f(x)\mathrm{d}x$ is given by

$$\sum_{i=0}^{n} A_i f(x_i),$$

where

$$A_i = \int_a^b \ell_i(x)\mathrm{d}x$$

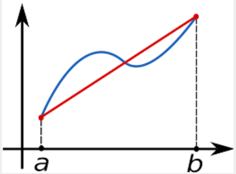
and

$$\ell_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

- Idea: replace f with something similar.
- In words, Newton-Cotes is a quadrature obtained from equally spaced Lagrange interpolation.
- Note that, once f is fixed, $f(x_i)$'s are data (they acts more like fixed numbers) while the "real" functions are $\ell_i(x)$'s.
- Notice that $f \mapsto \sum_{i=0}^{n} A_i f(x_i)$ is a linear mapping.

Trapezoidal rule

Trapezoidal rule $\longrightarrow$ Newton-Cotes with only two nodes a, b



$$\int_a^b f(x)dx \approx \frac{b-a}{2}[f(a)+f(b)]$$

The error term is

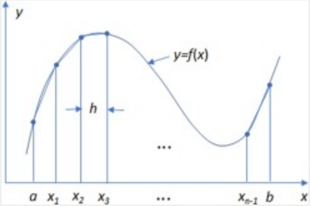
$$\frac{b-a}{2}[f(a)+f(b)] - \int_a^b f(x)dx = \frac{1}{12}(b-a)^3 f''(\xi)$$

We are not going to prove this version of error, but a more general version involving absolute value of error.

- Trapezoidal rule is exact for $f \in \Pi_1$
- For quadrature rules, the standard measure of accuracy is the degree of polynomials a formula is exact for.
- Most books has minus sign in the error. But I am sticking to the convention (error) $=$ (estimate) - (true): so negative error indicates underestimation (e.g., concave) and the positive overestimation (e.g., convex).

Trapezoidal rule and its composite version

Composite trapezoidal rule $\longrightarrow$ refine the interval into n subintervals and apply trapezoidal rule to each subinterval



$$\sum_{i=1}^{n}\int_{x_{i-1}}^{x_i} f(x)dx \approx \frac{1}{2}\sum_{i=1}^{n}(x_i - x_{i-1})\left[f(x_{i-1}) + f(x_i)\right]$$

If the refinement is uniform with $h = (b-a)/n$,

$$\int_a^b f(x)dx \approx \frac{h}{2}\left[f(a) + 2\sum_{i=1}^{n-1} f(a+ih) + f(b)\right]$$

- (**Exercise**) Show that the error term for the uniform refined case is $\frac{1}{12}(b-a)h^2 f''(\xi)$. (Hint: You may want to use the discrete version of mean value theorem for integral)

Simpson's Rule

Exercise: Derive a Newton-Cotes formula for functions defined on $[a, b]$ that is exact for $p \in \Pi_2$.

Guidelines

- Does this makes sense?
- Definition of Newton-Cotes?
- How many nodes to be chosen?
- What tool to use?

Simpson's Rule

Exercise: write a program that implement the composite Simpson's rule and test it.

- (minimum) input: a, b ($a < b$) (interval; two numbers), f (integrand; function handle), n (number of subintervals; an even integer)
- output: I (integral approximated by Simpson's rule)
- test: output $\int_0^{\pi/2} \sin x \mathrm{d}x$, $\int_{-1}^{1} f(x)\mathrm{d}x$, and $\int_0^2 e^{-x^2/2}\mathrm{d}x$, where $f(x) = \begin{cases} x^2/2 & (x \geq 0) \\ -x^2/2 & (x < 0) \end{cases}$.
- Confirm the convergence rate of the composite Simpson's rule when possible.
  - For each function, repeat the test while doubling n (number of subintervals) (e.g. $n = 16, 32, 64, 128, \cdots$)
  - Report a table of n, approximate integrals, error, $\log_2 e_k - \log_2 e_{k+1}$.
  - You need true value to compute the errors. You may want to use Wolfram alpha for such cases.

- This will be one of the options for computational HW2.

Change of interval

Goal: A quadrature formula for on $[a, b]$ for $\int_a^b f(x)\mathrm{d}x$.
What we have: $\int_c^d f(t)\mathrm{d}t \approx \sum_{i=0}^n A_i \ell_i$ on $[c, d]$.
Strategy: Change of variable: $x = \alpha t + \beta$ so that $c \mapsto a$ and $d \mapsto b$. Call such a map $x = \lambda(t)$.

Quadrature formula for different intervals

$$\int_a^b f(x)\mathrm{d}x \approx \frac{b-a}{d-c} \sum_{i=0}^n A_i f\left(\lambda\left(t_i\right)\right) = \frac{b-a}{d-c} \sum_{i=0}^n A_i f\left(x_i\right),$$

where $x_i = \lambda\left(t_i\right)$ are the new nodes on $[a, b]$ and t_i are the old nodes on $[c, d]$.

Proof.

Board work. □

In words,

- You can use a quadrature formula on an interval, say $[0, 1]$, for any other intervals.
- You change only the function values $f(x_i) = f(\lambda(t_i))$ while use the same weight A_i.
- You also need to adjust the total sum according to stretch or shrink (i.e., Jacobian $(b-a)/(d-c)$).

A naive but general error estimate

Intuition: If functions are similar, so are their integrals.

Quadrature error – single interval

For $h > 0$, Newton-Cotes formula of degree n on $[a, a+h]$ satisfies

$$\left| \int_a^{a+h} f(x)dx - \sum_{i=0}^{n} A_i f(x_i) \right| \le \frac{\hat{M}}{(n+1)!} h^{n+2},$$

where $\hat{M} := \max_{a \le x \le a+h} |f^{(n+1)}(x)|$ and $x_i = a + ih/n$ (i–th node; $i = 0, 1, 2, \cdots, n$).

Board work.

- Recall: $f(x) - p(x) = \frac{1}{(n+1)!} \cdot f^{(n+1)}(\xi_x) \prod_{i=0}^{n} (x - x_i)$
- **Subjective question**: Compare this result on the error with that of derivatives. Do you see any patterns or something notable?

A naive but general error estimate

Quadrature error – composite rules

Composite Newton-Cotes formula of degree n on $[a, b]$ with k uniform subintervals satisfies

$$\left| \int_a^b f(x)dx - \sum_{j=1}^{k} \sum_{i=0}^{n} A_i f\left(x_{j,i}\right) \right| \leq \frac{M(b-a)}{(n+1)!} h^{n+1}$$

where $M := \max_{a \leq x \leq b} |f^{(n+1)}(x)|$ and $x_{j,i} = a + (j-1)h + ih/n$ (i–th node in j–th subinterval; $i = 0, 1, 2, \cdots, n$ and $j = 1, 2, \cdots, k$).

Board work.

- Draw a picture to better decipher indices.

Numerical Differentiation and Integration

Gaussian Quadrature

Weighted quadrature

Sometimes, we need to integrate things w.r.t. some *density* or *weight* $w(t)\mathrm{d}t$.
Fortunately, what we have developed works similarly.

Weighted quadrature based on interpolation

Given $x_0, x_1, \cdots, x_n$, the following quadrature formula $\sum_{i=0}^{n} \tilde{A}_i f(x_i)$ is Π_n–exact, where

$$\tilde{A}_i = \int_a^b \ell_i(x) w(x) \mathrm{d}x,$$

Here, we are using only (n+1) degrees of freedom (for sensible $\tilde{A}_i$).
Question: can we do better by exploiting more degrees of freedom? If so, what to tune?

- w(t) is positive in the interior of the interval.
- Recall the fundamental interpolating polynomials

$$\ell_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}.$$

They satisfy $\ell_i(x_j) = \delta_{ij}$ (Kronecker delta)

Gaussian quadrature

Theorem

Let w be a positive weight function and $q \in \Pi_{n+1}$ be w–orthogonal to Π_n, that is, $\int_a^b q(x)p(x)w(x)dx = 0$ $(\forall p \in \Pi_n)$. If $x_0, x_1, \cdots, x_n$ are chosen to be the zeros of q, then the quadrature $\sum_{i=0}^{n} \tilde{A}_i f(x_i)$ is Π_{2n+1}–exact, where $\tilde{A}_i = \int_a^b \ell_i(x)w(x)\mathrm{d}x$.

Proof.

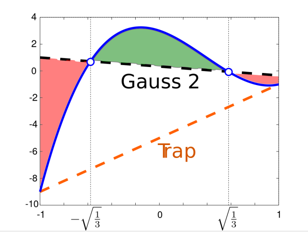
Board work. □

- From the degrees of freedom point of view, we've gained (n+1) higher accuracy by making (n+1) more sensible tuning.

Gaussian quadrature

Example of Gaussian quadrature

setting	quadrature formula
$n = 1$	$x_0 = -1/\sqrt{3},\ x_1 = 1/\sqrt{3}$
$[a, b] = [-1, 1]$	$\tilde{A}_0 = \tilde{A}_1 = 1$



- In theory, we can find arbitrarily accurate Gaussian formula.
- Once $[a, b]$, n, and $w(x)$ are chosen, everything is determined: $\Pi_n \rightarrow$ orthogonal polynomial $\rightarrow$ its roots $\rightarrow$ nodes x_i $\rightarrow$ Lagrange basis $\rightarrow$ weights $\tilde{A}_i$ $\rightarrow$ formula $\sum_{i=0}^{n} \tilde{A}_i f(x_i)$
- **Exercise**: confirm that this is accurate up to degree 3.

Properties of Orthogonal polynomials

Worries: What if the zeros are repeated? What if we have to plug in numbers outside [a,b]?

Theorem

Let $w \in C[a,b]$ be a positive weight function and $f \in C[a,b]$ be w–orthogonal to Π_n. Then, f changes sign at least $(n+1)$ times on (a,b).

Proof.

Board work. □

Corollary

The nodes of Gaussian quadrature are all distinct and inside the interval of integral.

Error

Lemma

The coefficients of the Gaussian quadrature are positive and sum up to $\int_a^b w(x)\mathrm{d}x$.

Proof.

Board work. □

Theorem (Convergence of Gaussian quadrature)

If $f \in C[a, b]$*, then the Gaussian quadrature converges to the true integral as* $n \to \infty$ *(*$n+1$ *is the number of nodes). That is,* $\sum_{i=0}^{n} \tilde{A}_{n,i} f(x_{n,i}) \to \int_a^b f(x)w(x)\mathrm{d}x$.

Proof.

Board work. □

- Convergence theorem is not of practical interest. But this shows why mathematicians care 'dense' subset and is a great exercise.

Error

Theorem (Error of the Gaussian quadrature)

If $f \in C^{2n}[a, b]$, then there is $\xi \in (a, b)$ such that

$$\int_a^b f(x)w(x)\mathrm{d}x - \sum_{i=0}^{n-1} \tilde{A}_i f(x_i) = \frac{1}{(2n)!} f^{(2n)}(\xi) \int_a^b \prod_{i=0}^{n} (x - x_i)^2 \,\mathrm{d}x$$

Proof.

We will not prove this. $\square$