16. Numerical integration

Summary of the previous lecture

- Interpolation
- Find a function $f(x)$ passing through points (x_i, y_i)
- Lagrange interpolation
- Barycentric Lagrange interpolation

Goals for today

- Numerical integration (“quadrature”)
- Approximating integrals
- Error analysis
- Conditioning

Need for numerical integration

- For a function f we often need to find definite integrals:

$$\int_a^b f(x)\,\mathsf{d}x = \int_a^b f$$

Need for numerical integration

- For a function f we often need to find definite integrals:

$$\int_a^b f(x)\,\mathrm{d}x = \int_a^b f$$

- Definite integrals are **more difficult** than derivatives
- In general there is *no* analytical solution, e.g. the **error function**

$$\mathrm{erf}(x) := \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\,\mathrm{d}t$$

Need for numerical integration

- For a function f we often need to find definite integrals:

$$\int_a^b f(x)\,\mathsf{d}x = \int_a^b f$$

- Definite integrals are **more difficult** than derivatives
- In general there is *no* analytical solution, e.g. the **error function**

$$\mathsf{erf}(x) := \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\,\mathsf{d}t$$

- Hence numerical integration is of paramount importance

Numerical integration problem

- The numerical integration problem:
- Given an input *function* f
  Calculate $I_{a,b}(f) := \int_a^b f(x)\,\mathrm{d}x$

Numerical integration problem

- The numerical integration problem:
- Given an input *function* f
  Calculate $I_{a,b}(f) := \int_a^b f(x)\,\mathrm{d}x$

- The problem is $\mathcal{Y} = \Phi(\mathcal{X})$ with

$$\mathcal{X} = f; \quad \Phi = \int; \quad \mathcal{Y} = \int f$$

Collaboration I

Simplest versions of numerical integration

1. What does $\int_a^b f$ represent geometrically?
2. What is the simplest approximation you could take?
3. How could you improve on that approximation?

Simplest case

- What is the simplest approximation of f?

Simplest case

- What is the simplest approximation of f?
- Approximate f using **rectangles** – rectangular rule
- As in Riemann integration

Simplest case

- What is the simplest approximation of f?
- Approximate f using **rectangles** – rectangular rule
- As in Riemann integration

- Split $[a, b]$ into N intervals (or **panels**) of length $h = \frac{b-a}{N}$
- Take **nodes** $x_k := a + k\,h$ (with $x_0 = a$ and $x_N = b$)

Rectangular rule II

- Approximate f by a piecewise-constant function p
- Choose value of p for each subinterval X_k

Rectangular rule II

- Approximate f by a piecewise-constant function p
- Choose value of p for each subinterval X_k

- e.g. $p(x) = f(x_k)$ for $x \in X_k := [x_k, x_{k+1})$

Rectangular rule II

- Approximate f by a piecewise-constant function p
- Choose value of p for each subinterval X_k

- e.g. $p(x) = f(x_k)$ for $x \in X_k := [x_k, x_{k+1})$
- So $p(x) = \sum_k f(x_k)\, \mathbb{1}_{X_k}(x)$
- Where $\mathbb{1}_{X_k}$ is **indicator function** of set

  = 1 if $x \in X_k$ and 0 if not

Rectangular rule III

- Area A_k of kth rectangle is $hf(x_k)$
- So $I(f) \simeq A(f,h) := h\sum_k f(x_k)$

Rectangular rule III

- Area A_k of kth rectangle is $hf(x_k)$
- So $I(f) \simeq A(f,h) := h\sum_k f(x_k)$

- Weights $w_k = h$ except $w_N = 0$

Collaboration II

How good is the rectangular rule?

Suppose we approximate $I(f)$ with $A(f, h)$, approximating f with a piecewise constant function.

1. How can we measure how good the approximation is?
2. Which mathematical tool could you use to find how this varies?
3. What do you obtain?

Error of rectangular rule

- How good is the rectangular rule?
- We want to calculate the **error**

$$E(h) := |A(f, h) - I(f)|$$

as a function of h, as $h \to 0$

Error of rectangular rule

- How good is the rectangular rule?
- We want to calculate the **error**

$$E(h) := |A(f, h) - I(f)|$$

  as a function of h, as $h \to 0$

- For small h the function is nearly constant in each X_k

Error of rectangular rule

- How good is the rectangular rule?
- We want to calculate the **error**

$$E(h) := |A(f, h) - I(f)|$$

as a function of h, as $h \to 0$

- For small h the function is nearly constant in each X_k
- Then it makes sense to model the function using a **Taylor expansion** around an end-point:

$$f(x) = f(x_k) + (x - x_k)\, f'(\xi_k) \quad \text{for } x \in X_k$$

Error of rectangular rule II

- $f(x) = f(x_k) + (x - x_k)\, f'(\xi_k)$ for $x, \xi_k \in X_k$

Error of rectangular rule II

- $f(x) = f(x_k) + (x - x_k)\, f'(\xi_k)$ for $x, \xi_k \in X_k$
- Suppose $|f'|$ is bounded in X_k by M_k
- Then for $x \in X_k$ we have:

$$|f(x) - p(x)| = |f(x) - f(x_k)|$$

Error of rectangular rule II

- $f(x) = f(x_k) + (x - x_k)\, f'(\xi_k)$ for $x, \xi_k \in X_k$
- Suppose $|f'|$ is bounded in X_k by M_k
- Then for $x \in X_k$ we have:

$$|f(x) - p(x)| = |f(x) - f(x_k)|$$

$$= |(x - x_k)\, f'(\xi_k)|$$

Error of rectangular rule II

- $f(x) = f(x_k) + (x - x_k)\, f'(\xi_k)$ for $x, \xi_k \in X_k$
- Suppose $|f'|$ is bounded in X_k by M_k
- Then for $x \in X_k$ we have:

$$|f(x) - p(x)| = |f(x) - f(x_k)|$$

$$= |(x - x_k)\, f'(\xi_k)|$$

$$\leq M_k h$$

Error of rectangular rule III

- Thus $|f(x) - p(x)| \leq Mh$ for x in X_k
- So $E_k := |\int_{X_k} (f - p)| \leq \int_{X_k} Mh \leq Mh^2$

Error of rectangular rule III

- Thus $|f(x) - p(x)| \leq Mh$ for x in X_k
- So $E_k := |\int_{X_k} (f - p)| \leq \int_{X_k} Mh \leq Mh^2$
- We have $N \sim 1/h$ subintervals
- So global error in integral is $E(f, h) = \sum_k E_k$

$$E(f, h) = \int_a^b [f(x) - p(x)] = \mathcal{O}(h)$$

Collaboration III

Improving the approximation

1. How can we get a better method?
2. How small an error do you expect?
3. Is there a way of minimising the error even more?

Improving the approximation

- How can we improve this, i.e. reduce the error?

Improving the approximation

- How can we improve this, i.e. reduce the error?
- We need to *use a better approximation*!

Improving the approximation

- How can we improve this, i.e. reduce the error?
- We need to *use a better approximation*!
- Try modelling the function with a piecewise-linear function instead of piecewise-constant

Improving the approximation

- How can we improve this, i.e. reduce the error?
- We need to *use a better approximation*!
- Try modelling the function with a piecewise-linear function instead of piecewise-constant

- Interpolate using Lagrange cardinal polynomials:

Improving the approximation

- How can we improve this, i.e. reduce the error?
- We need to *use a better approximation*!
- Try modelling the function with a piecewise-linear function instead of piecewise-constant

- Interpolate using Lagrange cardinal polynomials:
- $p_1(x) = \ell_0(x) f(a) + \ell_1(x) f(b)$

$$p_1(x) = \frac{x-b}{a-b}\, f(a) + \frac{x-a}{b-a}\, f(b)$$

Trapezium rule

- Now integrate p_1:

Trapezium rule

- Now integrate p_1:
- $\int_a^b p_1(x)\,dx = f(a)\int \ell_0 + f(b)\int \ell_1$

Trapezium rule

- Now integrate p_1:
- $\int_a^b p_1(x)\,dx = f(a)\int \ell_0 + f(b)\int \ell_1$
- We get $\int_a^b p_1(x)\,dx = \frac{1}{2}(b-a)\left[f(a)+f(b)\right]$

Trapezium rule

- Now integrate p_1:
- $\int_a^b p_1(x)\,dx = f(a)\int \ell_0 + f(b)\int \ell_1$
- We get $\int_a^b p_1(x)\,dx = \frac{1}{2}(b-a)\,[f(a)+f(b)]$
- A_k is now the area of a **trapezium**:

$$A_k = \frac{h}{2}[f(x_k) + f(x_{k+1})]$$

- The total area is then

$$A(h) = h[\tfrac{1}{2}f(a) + f(x_1) + \cdots + f(x_{k-1}) + \tfrac{1}{2}f(b)]$$

Numerical integration: nodes and weights

- We need to evaluate the function f at **nodes** x_k
- $a = x_0 < x_1 < \cdots < x_n = b$

Numerical integration: nodes and weights

- We need to evaluate the function f at **nodes** x_k
- $a = x_0 < x_1 < \cdots < x_n = b$
- In general we want an approximation like

$$\int_a^b f(x) \simeq \sum_k w_k f(x_k)$$

- The formula should work for “any” f

Numerical integration: nodes and weights

- We need to evaluate the function f at **nodes** x_k
- $a = x_0 < x_1 < \cdots < x_n = b$
- In general we want an approximation like

$$\int_a^b f(x) \simeq \sum_k w_k\, f(x_k)$$

- The formula should work for "any" f
- So the **weights** w_k should be **independent** of f

Numerical integration: nodes and weights

- We need to evaluate the function f at **nodes** x_k
- $a = x_0 < x_1 < \cdots < x_n = b$
- In general we want an approximation like

$$\int_a^b f(x) \simeq \sum_k w_k\, f(x_k)$$

- The formula should work for “any” f
- So the **weights** w_k should be **independent** of f
- Nodes may be given, or we may *choose* nodes and weights

Numerical integration: nodes and weights

- We need to evaluate the function f at **nodes** x_k
- $a = x_0 < x_1 < \cdots < x_n = b$
- In general we want an approximation like

$$\int_a^b f(x) \simeq \sum_k w_k\, f(x_k)$$

- The formula should work for "any" f
- So the **weights** w_k should be **independent** of f
- Nodes may be given, or we may *choose* nodes and weights
- Note: $\int$ and this approximation are both **linear** operators

Interpolation: Newton–Cotes

- Rectangular and trapezium rules: *interpolate* then *integrate*

Interpolation: Newton–Cotes

- Rectangular and trapezium rules: *interpolate* then *integrate*
- Generalise: **Newton–Cotes** rules (equally-spaced nodes)

Interpolation: Newton–Cotes

- Rectangular and trapezium rules: *interpolate* then *integrate*
- Generalise: **Newton–Cotes** rules (equally-spaced nodes)

- Error when interpolate f by a degree-n polynomial p_n is

$$f(x) = p_n(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}\pi_n(x)$$

where $\pi_n(x) := \prod_{k=0}^{n}(x - x_k)$ – degree $n+1$

Interpolation: Newton–Cotes

- Rectangular and trapezium rules: *interpolate* then *integrate*
- Generalise: **Newton–Cotes** rules (equally-spaced nodes)

- Error when interpolate f by a degree-n polynomial p_n is

$$f(x) = p_n(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}\pi_n(x)$$

where $\pi_n(x) := \prod_{k=0}^{n}(x - x_k)$ – degree $n+1$

- Note that the interpolation error $= 0$ at each node!

Error for Newton–Cotes rules

- Integrating the interpolant gives
- $|\int f - \int p_n| \leq \frac{M_{n+1}}{(n+1)!} \int |\pi_n|$

  where M_{n+1} is a bound for $|f^{(n+1)}|$ on $[a, b]$

Error for Newton–Cotes rules

- Integrating the interpolant gives
- $|\int f - \int p_n| \leq \frac{M_{n+1}}{(n+1)!} \int |\pi_n|$

  where M_{n+1} is a bound for $|f^{(n+1)}|$ on $[a, b]$

- Since π_n is of degree $n + 1$, integrating it gives $\mathcal{O}(h^{n+2})$
- So global error is $\mathcal{O}(h^{n+1})$

Error for Newton–Cotes rules

- Integrating the interpolant gives
- $\left| \int f - \int p_n \right| \leq \frac{M_{n+1}}{(n+1)!} \int |\pi_n|$

  where M_{n+1} is a bound for $\left| f^{(n+1)} \right|$ on $[a, b]$

- Since π_n is of degree $n+1$, integrating it gives $\mathcal{O}(h^{n+2})$
- So global error is $\mathcal{O}(h^{n+1})$
- E.g. trapezium rule has error $\mathcal{O}(h^2)$

Conditioning of numerical integration

- What is the condition number of the integration problem?
- Input: f; output: $I(f) = \int_a^b f$

Conditioning of numerical integration

- What is the condition number of the integration problem?
- Input: f; output: $I(f) = \int_a^b f$
- Perturb the input function f by function Δf
- Then the output perturbation ΔI is

$$\Delta I = I(f + \Delta f) - I(f)$$

Conditioning of numerical integration

- What is the condition number of the integration problem?
- Input: f; output: $I(f) = \int_a^b f$
- Perturb the input function f by function Δf
- Then the output perturbation ΔI is

$$\Delta I = I(f + \Delta f) - I(f)$$

- So $\Delta I = I(\Delta f)$

Conditioning II

- We have $\Delta I = I(\Delta f)$
- So

$$|\Delta I| = \left| \int \Delta f \right| \le \int |\Delta f| =: \|\Delta f\|_1$$

- The relative error is then

$$\left| \frac{\Delta I}{I} \right| \le \frac{\|\Delta f\|_1}{|I|}$$

Conditioning III

- So the relative condition number is

$$\kappa = \frac{|\Delta I\ |/|I|}{\|\Delta f\|/\|f\|} = \frac{\|f\|_1}{|I|}$$

- Hence

$$\kappa = \frac{\int_a^b |f(x)|\,dx}{\left|\int_a^b f(x)\,dx\right|}$$

Conditioning III

- So the relative condition number is

$$\kappa = \frac{|\Delta I\ |/|I|}{\|\Delta f\|/\|f\|} = \frac{\|f\|_1}{|I|}$$

- Hence

$$\kappa = \frac{\int_a^b |f(x)|\, dx}{\left|\int_a^b f(x)\, dx\right|}$$

- We see that this is *ill-conditioned* when $|f|$ is large but $\int f$ is small
- I.e. when we integrate a **highly-oscillatory** function

Adaptivity

- So far we have always taken equally-spaced nodes

Adaptivity

- So far we have always taken equally-spaced nodes
- Why? If we have the choice of where to choose the nodes, we may be able to do better!

Adaptivity

- So far we have always taken equally-spaced nodes
- Why? If we have the choice of where to choose the nodes, we may be able to do better!

- Idea of adaptivity: Choose new nodes where it makes sense to do so

Adaptivity

- So far we have always taken equally-spaced nodes
- Why? If we have the choice of where to choose the nodes, we may be able to do better!

- Idea of adaptivity: Choose new nodes where it makes sense to do so
- I.e. where the function “behaves more badly”

Higher dimensions

- The problem of integrating e.g. $f : \mathbb{R}^2 \to \mathbb{R}$ is sometimes called **cubature**

Higher dimensions

- The problem of integrating e.g. $f : \mathbb{R}^2 \to \mathbb{R}$ is sometimes called **cubature**
- It is much more complicated than the 1D problem

Higher dimensions

- The problem of integrating e.g. $f : \mathbb{R}^2 \to \mathbb{R}$ is sometimes called **cubature**
- It is much more complicated than the 1D problem
- There are many more ways of dividing up 2D space

Contour integration

- Integrate a *complex*-valued function along a *curve* in $\mathbb{C}$:

$$\int_C f(z)\,\mathsf{d}z$$

Contour integration

- Integrate a *complex*-valued function along a *curve* in $\mathbb{C}$:

$$\int_C f(z)\,\mathsf{d}z$$

- **Parametrise** by a function $t \mapsto \gamma(t)$ with *image* C

Contour integration

- Integrate a *complex*-valued function along a *curve* in $\mathbb{C}$:

$$\int_C f(z)\,\mathsf{d}z$$

- **Parametrise** by a function $t \mapsto \gamma(t)$ with *image* C
- We define

$$\int_C f(z)\,\mathsf{d}z := \int_{t=0}^{1} f(\gamma(t))\,\gamma'(t)\,\mathsf{d}t$$

Contour integration

- Integrate a *complex*-valued function along a *curve* in $\mathbb{C}$:

$$\int_C f(z)\,\mathrm{d}z$$

- **Parametrise** by a function $t \mapsto \gamma(t)$ with *image* C
- We define

$$\int_C f(z)\,\mathrm{d}z := \int_{t=0}^{1} f(\gamma(t))\,\gamma'(t)\,\mathrm{d}t$$

- We can then carry out this integration *numerically*!
- Note that the result is independent of the parametrisation

Contour integration II

- Split up $f(\gamma(t))\gamma'(t)$ into its real and imaginary parts

Contour integration II

- Split up $f(\gamma(t))\gamma'(t)$ into its real and imaginary parts
- Integrate separately – now 1D integrals

Contour integration II

- Split up $f(\gamma(t))\gamma'(t)$ into its real and imaginary parts
- Integrate separately – now 1D integrals
- *Closed* curves are a particular case of importance in complex analysis
- e.g. $\gamma(t) = \exp(2\pi i t)$

Contour integration II

- Split up $f(\gamma(t))\gamma'(t)$ into its real and imaginary parts
- Integrate separately – now 1D integrals
- *Closed* curves are a particular case of importance in complex analysis
- e.g. $\gamma(t) = \exp(2\pi i t)$

- It turns out that the trapezium rule is surprisingly good for **periodic** functions like this

Summary

- Numerical integration approximates definite integral
- Interpolate then integrate
- Degree n polynomial leads to $\mathcal{O}(h^{n+1})$ error