Machine Learning and Photonics

Math Foundations: Numerical Integration

Ergun Simsek

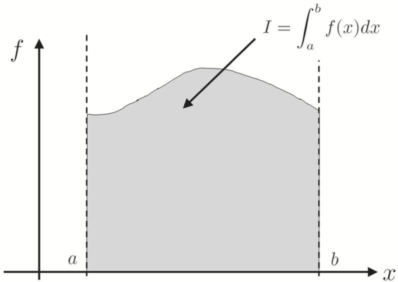
University of Maryland Baltimore County

simsek@umbc.edu

February 7, 2023

Integral of a function = Area under the curve.

- Given a function $f(x)$, we want to approximate the integral of $f(x)$ over the total **interval**, $[a, b]$.
- The interval has been discretized into a numeral grid, x, consisting of $n + 1$ points with spacing, $h = \frac{b-a}{n}$.



Riemann Integration

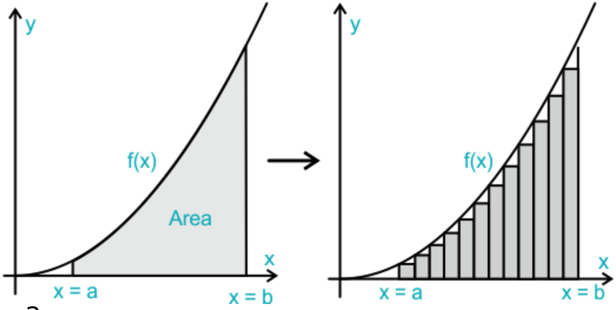
Summing the area of rectangles that are defined for each subinterval!

Using left end-points

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} hf(x_i),$$

Using right end-points

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} hf(x_i),$$



How accurate is Riemann Integration?

Riemann Integration: Accuracy

Let's rewrite the integral of $f(x)$ over an arbitrary subinterval in terms of the Taylor series of $f(x)$ around $a = x_i$

$$f(x) = f(x_i) + f'(x_i)(x - x_i) + \cdots$$

$$\int_{x_i}^{x_{i+1}} f(x)dx = \int_{x_i}^{x_{i+1}} \left(f(x_i) + f'(x_i)(x - x_i) + \cdots\right) dx$$

Since the integral distributes, we can rearrange the right side into the following form:

$$\int_{x_i}^{x_{i+1}} f(x_i)dx + \int_{x_i}^{x_{i+1}} f'(x_i)(x - x_i)dx + \cdots .$$

Solving each integral separately results in the approximation

$$\int_{x_i}^{x_{i+1}} f(x)dx = hf(x_i) + \frac{h^2}{2} f'(x_i) + \mathcal{O}(h^3),$$

Riemann Integration: Accuracy *(Cont...)*

$$\int_{x_i}^{x_{i+1}} f(x)dx = hf(x_i) + \mathcal{O}(h^2).$$

Since the $hf(x_i)$ term is our Riemann integral approximation for a single subinterval, the Riemann integral approximation over a single interval is $\mathcal{O}(h^2)$.

If we sum the $\mathcal{O}(h^2)$ error over the entire Riemann sum, we get $n\mathcal{O}(h^2)$.

Remember, $h = \frac{b-a}{n}$

So our total error becomes $\frac{b-a}{h}\mathcal{O}(h^2) = \mathcal{O}(h)$ over the whole interval.

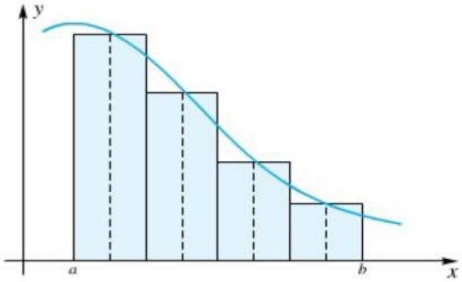
Thus the overall accuracy is $\mathcal{O}(h)$.

Mid-Point Rule

Let's take the rectangle height of the rectangle at each subinterval to be the function value at the midpoint between x_i and x_{i+1}

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} hf(\frac{x_{i+1}+x_i}{2}).$$

A mathematical proof will be provided in the lecture note but
why do you think the mid-point integration is $\mathcal{O}(h^2)$?



Example: $\int_0^{\pi} \sin(x)dx$

Use the left Riemann Integral, right Riemann Integral, and Midpoint Rule to approximate $\int_0^{\pi} \sin(x)dx$ with 11 evenly spaced grid points over the whole interval. Compare this value to the exact value of 2.

Example: $\int_0^{\pi} \sin(x)\,dx$

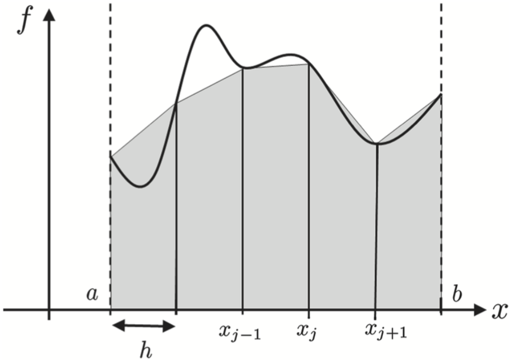
```python
import numpy as np
a, b, n = 0, np.pi, 11
h = (b - a) / (n - 1)
x = np.linspace(a, b, n)
f = np.sin(x)

I_riemannL = h * sum(f[:n-1])
err_riemannL = 2 - I_riemannL

I_riemannR = h * sum(f[1::])
err_riemannR = 2 - I_riemannR

I_mid = h * sum(np.sin((x[:n-1] + x[1:])/2))
err_mid = 2 - I_mid

print(I_riemannL)
print(err_riemannL)
print(I_riemannR)
print(err_riemannR)
print(I_mid)
print(err_mid)
```

```
1.9835235375094546
0.01647646249054535

1.9835235375094546
0.01647646249054535

2.0082484079079745
-0.008248407907974542
```

Trapezoid Rule

Sums the areas of the trapezoid to approximate the total integral

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} h\frac{f(x_i)+f(x_{i+1})}{2}.$$

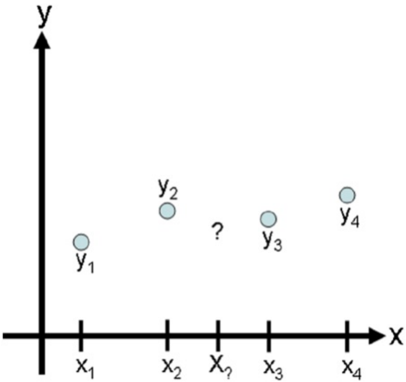


Notice that the Trapezoid Rule "double-counts" almost all the terms in the series (except the first and last ones)

You can use the Taylor Series and prove that Trapezoid integration is $\mathcal{O}(h^2)$.

A Brief Introduction to Interpolation

- Assume we have a data set consisting of independent data values, x_i, and dependent data values, y_i, where $i = 1, \ldots, n$.
- We would like to find an estimation function $\hat{y}(x)$ such that $\hat{y}(x_i) = y_i$ for every point in our data set.
- This means the estimation function goes through our data points.
- Given a new $x_?$, we can "interpolate" its function value using $\hat{y}(x_?)$.
- Here, $\hat{y}(x)$ is called an "interpolation function".



Lagrange Polynomials

Lagrange polynomial interpolation finds a single polynomial that goes through all the data points.
Lagrange polynomials, $P_i(x)$, are written as

$$P_i(x) = \prod_{j=1, j \neq i}^{n} \frac{x - x_j}{x_i - x_j},$$

and

$$L(x) = \sum_{i=1}^{n} y_i P_i(x).$$

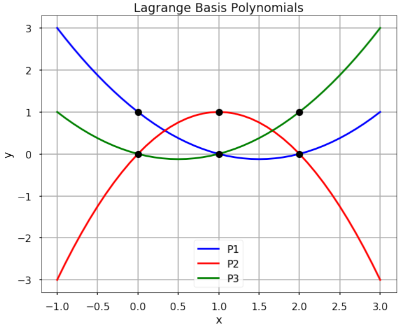
where, $\prod$ means the product of or multiply out.

Notice that by construction, $P_i(x)$ has the property that $P_i(x_j) = 1$ when $i = j$ and $P_i(x_j) = 0$ when $i \neq j$.

A Brief Introduction to Lagrange Interpolation

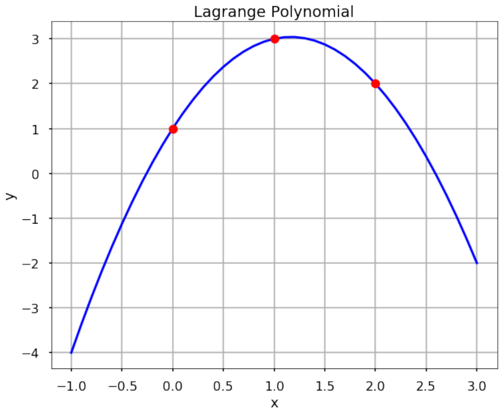
Let's find the Lagrange basis polynomials for the data set $x = [0, 1, 2]$ and $y = [1, 3, 2]$ and plot each polynomial and verify the property that $P_i(x_j) = 1$ when $i = j$ and $P_i(x_j) = 0$ when $i \neq j$.

$$
\begin{aligned}
P_1(x) &= \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} = \frac{(x-1)(x-2)}{(0-1)(0-2)} = \frac{1}{2}(x^2-3x+2), \\
P_2(x) &= \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} = \frac{(x-0)(x-2)}{(1-0)(1-2)} = -x^2+2x, \\
P_3(x) &= \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} = \frac{(x-0)(x-1)}{(2-0)(2-1)} = \frac{1}{2}(x^2-x).
\end{aligned}
$$



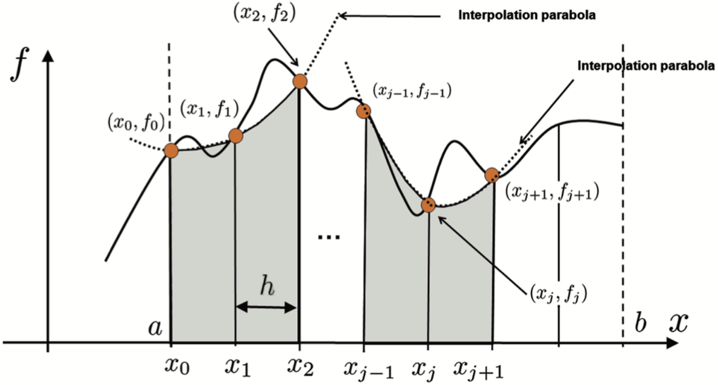
A Brief Introduction to Lagrange Interpolation

```
L = P1 + 3*P2 + 2*P3

fig = plt.figure(figsize = (10,8))
plt.plot(x_new, L(x_new), 'b', x, y,
    'ro')
plt.title('Lagrange Polynomial')
plt.grid()
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Simpson's Rule

Approximate the area under $f(x)$ over these two subintervals by fitting a quadratic polynomial through the points $(x_{i-1}, f(x_{i-1}))$, $(x_i, f(x_i))$, and $(x_{i+1}, f(x_{i+1}))$, which is a unique polynomial, and then integrate the quadratic exactly.



Simpson's Rule (Cont...)

The easiest way to implement the Simpson's rule is to use Lagrange polynomials. By applying the formula for constructing Lagrange polynomials we get the polynomial

$$\begin{aligned} P_i(x) &= f(x_{i-1})\frac{(x-x_i)(x-x_{i+1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})} + f(x_i)\frac{(x-x_{i-1})(x-x_{i+1})}{(x_i-x_{i-1})(x_i-x_{i+1})} \\ &\quad + f(x_{i+1})\frac{(x-x_{i-1})(x-x_i)}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)}, \end{aligned}$$

and with substitutions for h results in

$$P_i(x) = \frac{f(x_{i-1})}{2h^2}(x-x_i)(x-x_{i+1}) - \frac{f(x_i)}{h^2}(x-x_{i-1})(x-x_{i+1}) + \frac{f(x_{i+1})}{2h^2}(x-x_{i-1})(x-x_i).$$

With some algebra and manipulation, the integral of $P_i(x)$ over the 2 subintervals is

$$\int_{x_{i-1}}^{x_{i+1}} P_i(x)dx = \frac{h}{3}(f(x_{i-1}) + 4f(x_i) + f(x_{i+1}).$$

Simpson's Rule (Cont...)

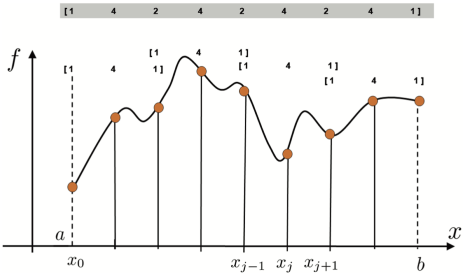
To approximate the integral over (a, b), we must sum the integrals of $P_i(x)$ over every *two* subintervals since $P_i(x)$ spans two subintervals.

Substituting $\frac{h}{3}(f(x_{i-1}) + 4f(x_i) + f(x_{i+1}))$ for the integral of $P_i(x)$ and regrouping the terms for efficiency leads to the formula

$$\int_a^b f(x)dx \approx \frac{h}{3}\left[f(x_0) + 4\left(\sum_{i=1, i \text{ odd}}^{n-1} f(x_i)\right) + 2\left(\sum_{i=2, i \text{ even}}^{n-2} f(x_i)\right) + f(x_n)\right].$$

Simpson's Rule (Cont...)

This regrouping can be illustrated as follows



WARNING! Note that to use Simpson's Rule, you **must** have an even number of intervals and, therefore, an odd number of grid points.

Simpson's rule is again $\mathcal{O}(h^2)$.

Example

Use Simpson's Rule to approximate $\int_0^{\pi} \sin(x)dx$ with 11 evenly spaced grid points over the whole interval. Compare this value to the exact value of 2.

```
import numpy as np
a = 0
b = np.pi
n = 11
h = (b - a) / (n - 1)
x = np.linspace(a, b, n)
f = np.sin(x)

I_simp = (h/3) * (f[0] + 2*sum(f[:n-2:2]) + 4*sum(f[1:n-1:2]) + f[n-1])
err_simp = 2 - I_simp

print(I_simp)
print(err_simp)
```

```
2.0001095173150043
-0.00010951731500430384
```

Example: Computing Integrals in Python

The *scipy.integrate* sub-package has several functions for computing integrals. The *trapz* takes as input arguments an array of function values f computed on a numerical grid x.

]:
```
import numpy as np
from scipy.integrate import trapz

a = 0
b = np.pi
n = 11
h = (b - a) / (n - 1)
x = np.linspace(a, b, n)
f = np.sin(x)

I_trapz = trapz(f,x)
I_trap = (h/2)*(f[0] + 2 * sum(f[1:n-1]) + f[n-1])

print(I_trapz)
print(I_trap)
```

```
1.9835235375094542
1.9835235375094546
```

Example-2: Computing Integrals in Python

Sometimes we want to know the approximated cumulative integral. That is, we want to know $F(X) = \int_{x_0}^{X} f(x)dx$. For this purpose, it is useful to use the *cumtrapz* function *cumsum*, which takes the same input arguments as *trapz*.

]:

```
from scipy.integrate import cumtrapz
import matplotlib.pyplot as plt

%matplotlib inline
plt.style.use('seaborn-poster')

x = np.arange(0, np.pi, 0.01)
F_exact = -np.cos(x)
F_approx = cumtrapz(np.sin(x), x)

plt.figure(figsize = (10,6))
plt.plot(x, F_exact)
plt.plot(x[1::], F_approx)
plt.grid()
plt.tight_layout()
plt.title('$F(x) = \int_0^{x} sin(y) dy$')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.legend(['Exact with Offset', 'Approx'])
plt.show()
```

Example-2: Computing Integrals in Python (Cont...)