PHYS 222 - Numerical Calculus

Fahed Abu Shaer

September 2023

1 Numerical Differentiation

One way to expand a function is Taylor's expansion. So, Taylor's expansion of a function around a point is,

$$f(x) = f(x_0) + (x - x_0)f^{'}(x_0) + \frac{(x - x_0)^2}{2!}f^{''}(x_0) + ...\frac{(x - x_0)^n}{n!}f^{(n)}(x_0) + ...$$

1.1 First Order Derivative

Let $h = x - x_0$, we truncate the expansion at h^2, this is only true if $h < 0$. So, the function becomes

$$f(x) = f(x_0) + (x - x_0)f^{'}(x_0) + O(h^2)$$

arranging the terms

$$f^{'}(x_0) = \frac{f(x) - f(x_0)}{x - x_0} - \frac{O(h^2)}{x - x_0}$$

knowing that $h = x - x_0$, the derivative becomes

$$f^{'}(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - O(h)$$

Considering the cases when x is shifted by h from the left and the right,

$$f(x-h) = f(x_0)+((x_0-h)-x_0)f^{'}(x_0)+\frac{((x_0 - h) - x_0)^2}{2!}f^{''}+O(h^3) = f(x_0)-hf^{'}(x_0)+\frac{h^2}{2}f^{''}+O(h^3)$$

$$f(x+h) = f(x_0)+((x_0+h)-x_0)f^{'}(x_0)+\frac{((x_0 + h) - x_0)^2}{2!}f^{''}+O(h^3) = f(x_0)+hf^{'}(x_0)+\frac{h^2}{2}f^{''}+O(h^3)$$

Now subtracting $f(x - h)$ from $f(x + h)$,

$$f(x + h) - f(x - h) = 2hf^{'}(x_0) + O(h^3)$$

Isolating the derivative and replacing x with x_0,

$$f^{'}(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h} + O(h^2)$$

To decrease the error to the order h^5, we shift x by $2h$ from the left and the right,

$$f(x_0+2h) = f(x_0) + 2hf^{'}(x_0) + \frac{4h^2}{2}f^{''}(x_0) + \frac{8h^3}{6}f^{'''}(x_0) + O(h^4)$$

$$f(x_0-2h) = f(x_0) - 2hf^{'}(x_0) + \frac{4h^2}{2}f^{''}(x_0) - \frac{8h^3}{6}f^{'''}(x_0) + O(h^4)$$

Subtracting $f(x_0-2h)$ from $f(x_0+2h)$,

$$f(x_0+2h) - f(x_0-2h) = 4hf^{'}(x_0) + \frac{8h^3}{3}f^{'''}(x_0) + O(h^5)$$

therefore

$$f(x_0+h) - f(x_0-h) = 2hf^{'}(x_0) + \frac{h^3}{3}f^{'''}(x_0) + O(h^5)$$

subtract them to get rid of the the third derivative and we get,

$$8f(x_0+h) - 8f(x_0-h) - f(x_0+2h) + f(x_0-2h) = 16hf^{'}(x_0) - 4hf^{'}(x_0) - \frac{8h^3}{3}f^{'''}(x_0) + \frac{8h^3}{3}f^{'''}(x_0) - O(h^5)$$

$$8f(x_0+h) - 8f(x_0-h) - f(x_0+2h) + f(x_0-2h) = 12hf^{'}(x_0) - O(h^5)$$

$$f^{'}(x_0) = \frac{1}{12h}(f(x_0-2h) - 8f(x_0-h) + 8f(x_0+h) - f(x_0+2h) + O(h^4)$$

1.2 Second Order Derivative

Consider the following expansions of the function,

$$f(x-h) = f(x_0) - hf^{'}(x_0) + \frac{h^2}{2!}f^{''} + O(h^3)$$

$$f(x+h) = f(x_0) + hf^{'}(x_0) + \frac{h^2}{2}f^{''} + O(h^3)$$

we add $f(x_0-h)$ and $f(x_0+h)$,

$$f(x_0+h) + f(x_0-h) = 2f(x_0) + h^2f^{''}(x_0) + O(h^4)$$

isolating the second order derivative,

$$f^{''}(x_0) = \frac{f(x_0-h) - 2f(x_0) + f(x_0+h)}{h^2} + O(h^2)$$

The second order derivative can be transformed into an addition of a diagonal matrix, D, that denotes how many neighbours we have, and a adjacency matrix,

A, that denotes the neighbours and connections.

$$D = \frac{1}{h^2}\begin{pmatrix} ... & ... & ... & ... & ... & ... & ... & ... \\ ... & 0 & 1 & -2 & 1 & 0 & ... & ... \\ ... & 0 & 0 & 1 & -2 & 1 & 0 & ... \\ ... & ... & ... & ... & ... & ... & ... & ... \end{pmatrix}$$

$$A = \begin{pmatrix} ... \\ ... \\ f(x_0 - h) \\ f(x_0) \\ f(x_0 + h) \\ ... \\ ... \end{pmatrix}$$

Therefore,

$$\nabla^2 = -D + A$$

2 Numerical Integration

To obtain the numerical value of the integral, defined in the region $[a, b]$, $I = \int_a^b f(x)dx$, there are two good methods and rules: trapezoidal rule and Simpson's rule.

2.1 Trapezoidal Rule

Trapezoidal rule is a rule that evaluates the area under the curves by dividing the total area into smaller trapezoids, unlike the Riemann Sum which uses rectangles, and summing over the areas of the trapezoids to estimate the area under the curve.
Let f(x) be a continuous function, and divide the area under the curve into trapezoid of height Δx. Generally, the area of a trapezoid is $A = \frac{h(b1+b2)}{2}$. So, the integral of the function $f(x)$ can be approximated to be

$$I = \int_{x_0}^{x_n} f(x)dx \approx (f(x_0)+f(x_1))\frac{\Delta x}{2}+(f(x_1)+f(x_2))\frac{\Delta x}{2}+...+(f(x_{n-1})+f(x_n))\frac{\Delta x}{2}$$

$$I \approx \sum_{i=1}^{n}(f(x_{i-1}) + f(x_i))\frac{\Delta x}{2}$$

The smaller the height of the trapezoid, the more accurate the integral is, and this was shown in the simulation that I ran where I approximated the integral of $f(x) = x^2$ in the interval $[-10, 10]$, $\int_{-10}^{10} x^2 dx$ with different Δx. This is the code I wrote,

```
def trap_integral(x, y):
    h = x[1] - x[0]
    I = 0
    for i in range(len(x)-1):
        I += (y[i] + y[i+1]) * h / 2
    return I

n = 10
I = np.empty(n)
h = np.empty(n)
for i in range(1, n + 1):
    x = np.linspace(-10, 10, 10 * i)
    y = x ** 2
    h[i-1] = x[1] - x[0]
    I[i-1] = trap_integral(x, y)
plt.plot(h, I)
plt.scatter(h, I)
plt.axis([h[0], h[n-1] - 0.1, I[n-1] - 2, I[0] + 3])
plt.ylabel("Numerical Integral")
plt.xlabel("Distance between two points")
plt.show()
print(h, "\n", I)
```

The values of the integral as a function of the distancing between two points are,

Δx	Integral Approximation
2.22	683.13
1.05	670.36
0.69	668.25
0.51	667.54
0.41	667.21
0.34	667.04
0.29	666.95
0.25	666.88
0.22	666.83
0.20	666.80

I plotted the integral approximation as a function of Δx,

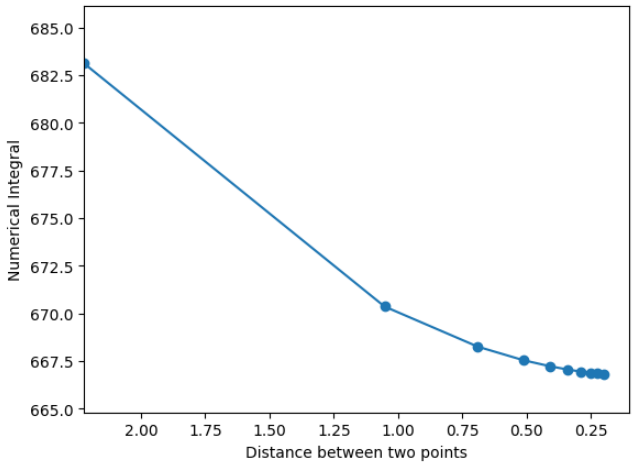

Figure 1: Integral Approximation as a function of Δx

We can clearly see that the approximation converges to a value around 666.8.
Then, I solved the integral analytically,

$$I = \int_{-10}^{10} x^2 dx = \left[\frac{x^3}{3}\right]_{-10}^{10} = \frac{(10)^3}{3} - \frac{(-10)^3}{3} = \frac{1000}{3} + \frac{1000}{3} = \frac{2000}{3} \approx 666.7$$

Therefore, the numerical trapezoidal approximations gives us results that are very close to the analytic results, especially when the height of the trapezoid is very small.

2.2 Simpson's Rule

Another way of approximating an integral numerically is using Simpson's rule. It is derived first by expanding the function using Taylor's expansion about the midpoint of the interval,

$$\begin{aligned} I &= \int_{x_0}^{x_2} f(x)dx \\ &= \int_{x_0}^{x_2} \left(f(x_1) + (x - x_1)f'(x_1) + \frac{(x - x_1)^2}{2} f''(x_1) + \frac{(x - x_1)^3}{6} f'''(x_1) \right. \\ &\qquad \left. + \frac{(x - x_1)^4}{24} f^{(4)}(x_1) + O(h^5) \right) dx \end{aligned}$$

then, we integrate each component of the expansion,

$$\int_{x_0}^{x_2} f(x)dx = [xf(x_1)]_{x_0}^{x_2} + \left[\frac{(x-x_1)^2}{2}f^{'}(x_1)\right]_{x_0}^{x_2} + \left[\frac{(x-x_1)^3}{6}f^{''}(x_1)\right]_{x_0}^{x_2} + \left[\frac{(x-x_1)^4}{24}f^{'''}(x_1)\right]_{x_0}^{x_2} + \left[\frac{(x-x_1)^5}{120}f^{(4)}(x_1)\right]_{x_0}^{x_2} + O(h^5)$$

evaluating the integrals,

$$\int_{x_0}^{x_2} f(x)dx = (x_2-x_0)f(x_1)+0+\frac{(x_2-x_0)^3}{6}f^{''}(x_1)+0+\frac{(x_2-x_0)^5}{120}f^{(4)}(x_1)+O(h^5)$$

let $x_2 - x_0 = 2h$, then

$$\int_{x_0}^{x_2} f(x)dx = 2hf(x_1) + \frac{2h^3}{6}f^{''}(x_1) + \frac{2h^5}{120}f^{(4)}(x_1) + O(h^5)$$

substituting the approximation of the second order derivative,

$$\int_{x_0}^{x_2} f(x)dx = 2hf(x_1) + \frac{h^3}{3}\left(\frac{f(x_0)-2f(x_1)+f(x_2)}{h^2}\right) + O(h^5) = \frac{h}{3}f(x_0) + \frac{4h}{3}f(x_1) + \frac{h}{3}f(x_2) + O(h^5)$$

I constructed a function that would compute numerically for me an integral, using Simpson's Rule, given the coordinates, the distance between each one, and the order of the interpolated function. The function is,

```
def SimpsonsRule(x, y, h, n):
    b = least_square_approximation(x, y, n)
    l = fitpolynomial([x[0] - h], b, n)
    I = ((4/3) * h * y[0]) + ((h/3) * l) + ((h/3) * y[1])
    for i in range(1, len(y) - 1, 2):
        I += ((4/3) * h * y[i]) + ((h/3) * y[i-1]) + ((h/3) * y[i+1])
    k = fitpolynomial([x[len(x) - 1] + h], b, n)
    I += ((4/3) * h * y[len(y) - 1]) + ((h/3)* y[len(y) - 2]) + ((h/3) * k)
    return I
```

and I applied this function on the integral $\int_{-10}^{10} x^2 dx$ with varying h, computed the difference between the analytical solution $I = 666.67$ and the numerical solution, and I plotted the integral as a function of h,

```
l = 10000
I = []
h = []

x1 = np.linspace(-10,10,10)
y1 = x1**2
d1 = x1[1] - x1[0]
```

```
Inte = SimpsonsRule(x1, y1, d1, 2)
error = np.abs(666.67 - Inte[0])
print("h = ", d1, " I = ", Inte[0], " Error = ", error)

for i in range(510, l + 1, 500):
    n = 10 * i
    x = np.linspace(-10, 10, n)
    y = x**2
    d = x[1] - x[0]
    Int = SimpsonsRule(x, y, d, 2)
    h.append(d)
    I.append(Int[0])
    error = np.abs(666.67 - Int[0])
    print("h = ", d, " I = ", Int[0], " Error = ", error)
plt.plot(h, I)
plt.scatter(h, I)
plt.axis([h[0], h[-1], I[-1] - 2, I[0] + 3])
plt.ylabel("Numerical Integral")
plt.xlabel("Distance between two points")
plt.show()
print(h, "\n", I)
```

The values of the integral, the distance between then h and the errors are,

h	Integral Approximation	Error
2.222	1393.68	727.01
0.080	690.82	24.15
0.011	670.09	3.42
0.007	668.84	2.18
0.003	667.80	1.14
0.0025	667.44	0.77
0.0019	667.25	0.58
0.0016	667.14	0.47
0.0010	666.96	0.29
0.0007	666.88	0.21
0.00044	666.79	0.12
0.00036	666.77	0.1
0.00028	666.75	0.08
0.00022	666.73	0.06
0.00020	666.72	0.05

and the graph of the numerical integral approximation as a function of h in the range $[0.02, 0.0002]$ is,

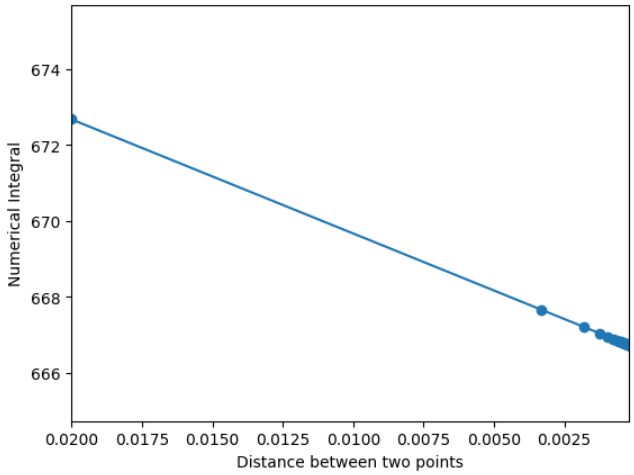

Figure 2: Numerical Integral Approximation of $\int_{-10}^{10} x^2 dx$ as a function of distance between two consecutive points h

As we can see, when h was greater than 1, the method gave us a completely wrong answer, however as we decreased the distance between two consecutive points, the numerical approximation becomes closer and closer to the analytic solution $I = 666.67$.