Ray_hw3

February 29, 2024

1 Exercise 1

$x-h,\ x,\ x+h$

$x_1 = x$

$x_2 = x-h$

$x_3 = x+h$

$$\frac{f[x_2,x_3]-f[x_1,x_2]}{x_3-x_1} = \frac{\frac{f(x_3)-f(x_2)}{x_3-x_2} - \frac{f(x_2)-f(x_1)}{x_2-x_1}}{x_3-x_1}$$

$$(y-x_1)(y-x_2) = [y^2-(x_1+x_2)y+x_1x_2]$$

$$f_1(y) = f[x_1] + f[x_1,x_2](y-x_1) + f[x_1,x_2,x_3](y-x_1)(y-x_2)$$

$$= f(x_1) + \frac{f(x_1)-f(x_2)}{x_1-x_2}(y-x_1) + $$

$$= f(x) + \frac{f(x)-f(x-h)}{h}(y-x_1) + \frac{\frac{f(x+h)-f(x-h)}{2h} + \frac{2(f(x-h)-f(x))}{2h}}{h}$$

$$f'(y)\Big|_{y=x} = \frac{f(x)-f(x-h)}{h} + \frac{f(x+h)-f(x-h)+2f(x-h)-2f(x)}{2h^2}$$

~~$[y^2-(x_1+x_2)y+x_1x_2]$~~

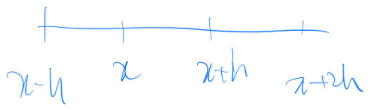
$(2y-(x_1+x_2))$

$2x - 2x+h$

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

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

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



$x_1 = x$

$x_2 = x-h$

$x_3 = x+h$

$x_4 = x+2h$

$$f_2(y) = f[x_1] + f[x_1\ x_2](y-x_1) + f[x_1\ x_2\ x_3](y-x_1)(y-x_2) + f[x_1\ x_2\ x_3\ x_4](y-x_1)(y-x_2)(y-x_3)$$

We see $f_2(y) = f_1(y) + f[x_1\ x_2\ x_3\ x_4](y-x_1)(y-x_2)(y-x_3)$

Thus, : to find $f_2'(y)\big|_{y=x_1}$, we only need to calculate the second part

$$\left[(y-x_1)(y-x_2)(y-x_3)\right]'\Big|_{y=x_1} = -2(x_1+x_2+x_3)y + x_1(x_2+x_3) + x_2x_3 + 3y^2$$

$$= -2(x_1+x_2+x_3)x_1 + x_1(x_2+x_3) + x_2x_3 + 3x_1^2$$

$$= -2x_1^2 \underline{-2x_2x_1} - 2x_3x_1 \underline{+ x_1x_2} + x_1x_3 + x_2x_3 + 3x_1^2$$

$$= x_1^2 - x_1x_2 - x_3x_1 + x_2x_3$$

$$= x^2 - x(x-h) - (x+h)x + (x-h)(x+h)$$

$$= \underline{x^2 - x^2} + \underline{xh} - x^2 - \underline{xh} + x^2 - h^2$$

$$= -h^2$$

$$f(x_3) - \underline{f(x_2)} - 2f(x_1) \underline{+2f(x_2)}$$
$$+f(x_2)$$

$x_1 = x \qquad f(x_1)$

$x_2 = x-h \qquad f(x_2)$

$x_3 = x+h \qquad f(x_3)$

$x_4 = x+2h \qquad f(x_4)$

$$[x_1,x_2]\quad \frac{f(x_1)-f(x_2)}{x_1-x_2} = \frac{f(x_1)-f(x_2)}{h}$$

$$[x_2,x_3]\quad \frac{f(x_2)-f(x_3)}{x_2-x_2} = \frac{f(x_3)-f(x_2)}{2h}$$

$$[x_3,x_4]\quad \frac{f(x_3)-f(x_4)}{x_3-x_4} = \frac{f(x_4)-f(x_3)}{h}$$

$$[x_1,x_2,x_3]\quad \frac{f[x_2,x_3]-f[x_1,x_2]}{x_3-x_1} = \frac{\frac{f(x_3)-f(x_2)}{2h} - \frac{2f(x_1)-2f(x_2)}{2h}}{h} = \frac{f(x_3)+f(x_2)-2f(x_1)}{2h^2}$$

$$[x_2,x_3,x_4]\quad \frac{f[x_3\ x_4]-f[x_2\ x_3]}{x_4-x_2} = \frac{\frac{2f(x_4)-2f(x_3)}{2h} + \frac{-f(x_3)+f(x_2)}{2h}}{3h} = \frac{2f(x_4)-3f(x_3)+f(x_2)}{6h^2}$$

$$f[x_1\ x_2\ x_3\ x_4] = \frac{f[x_2,x_3\ x_4]-f[x_1,x_2,x_3]}{x_4-x_1} = \frac{\frac{2f(x_4)-3f(x_3)+f(x_2)}{6h^2} - \frac{f(x_3)+f(x_2)-2f(x_1)}{2h^2}}{2h}$$

$$= \frac{2f(x_4)-3f(x_3)+f(x_2)-3f(x_3)-3f(x_2)+6f(x_1)}{2h \times 6h^2}$$

$$= \frac{2f(x_4)-6f(x_3)-2f(x_2)+6f(x_1)}{12h^3}$$

Thus

$$\left[f[x_1 x_2 x_3 x_4] (y-x_1)(y-x_2)(y-x_3) \right]' \Big|_{y=x_1} = \frac{2f(x_4) - 6f(x_3) - 2f(x_2) + 6f(x_1)}{12h^3} \cdot (-h^2)$$

$$= \frac{2f(x_4) - 6f(x_3) - 2f(x_2) + 6f(x_1)}{-12h}$$

$$f_2'(y) \Big|_{y=x_1} = \frac{f(x+h) - f(x-h)}{2h} - \frac{2f(x_4) - 6f(x_3) - 2f(x_2) + 6f(x_1)}{12h}$$

$$= \frac{6f(x_3) - 6f(x_2) - 2f(x_4) + 6f(x_3) + 2f(x_2) - 6f(x_1)}{12h}$$

$$= \frac{12f(x_3) - 4f(x_2) - 6f(x_1) - 2f(x_4)}{12h}$$

$$= \frac{6f(x_3) - 2f(x_2) - 3f(x_1) - f(x_4)}{6h} = \frac{6f(x+h) - 2f(x-h) - 3f(x) - f(x+2h)}{6h}$$

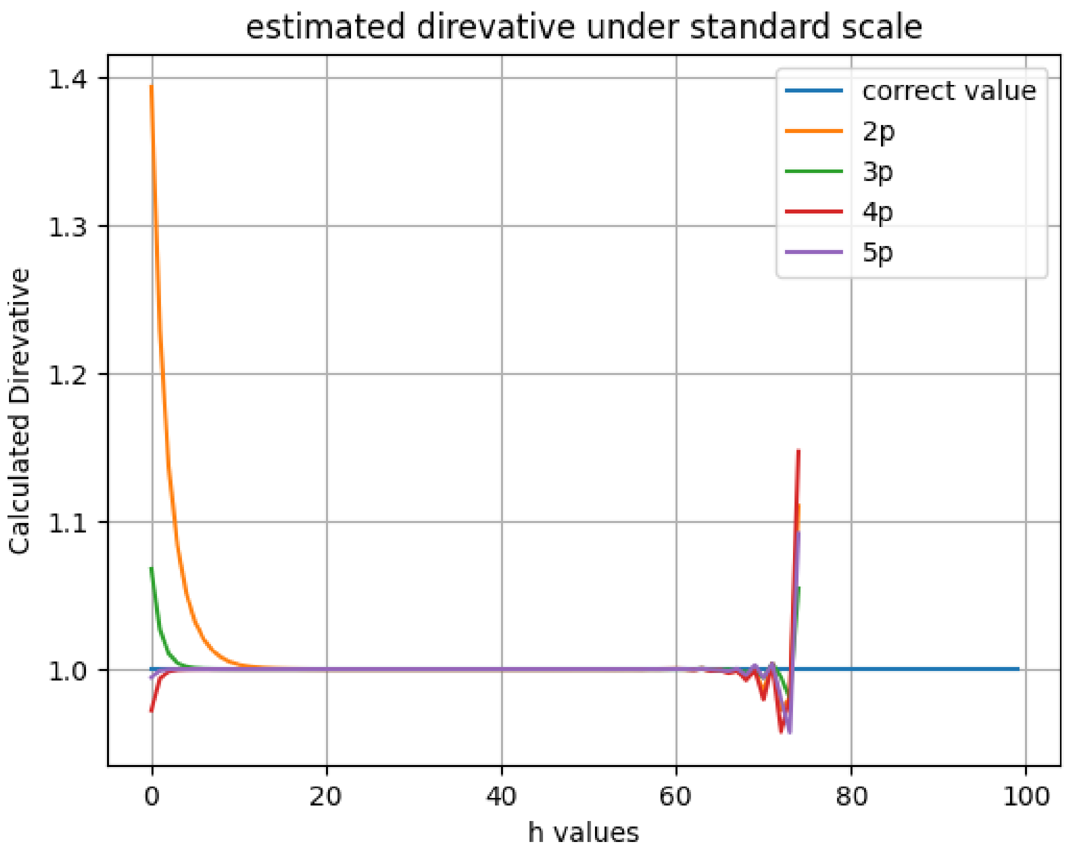
2 Exercise 2

[1]:
```python
# Set up
from pylab import *
import math
```

[2]:
```python
def calc_diff_2p(f, x, h):
    return (f(x + h) - f(x)) / h

def calc_diff_3p(f, x, h):
    return (f(x + h) - f(x - h)) / (2 * h)

def calc_diff_4p(f, x, h):
    return (6 * f(x + h) - 2 * f(x - h) - 3 * f(x) - f(x + 2 * h)) / (6 * h)

def calc_diff_5p(f, x, h):
    return (f(x - 2*h) - 8 * f(x - h) + 8 * f(x + h) - f(x + 2 * h)) / (12 * h)
```

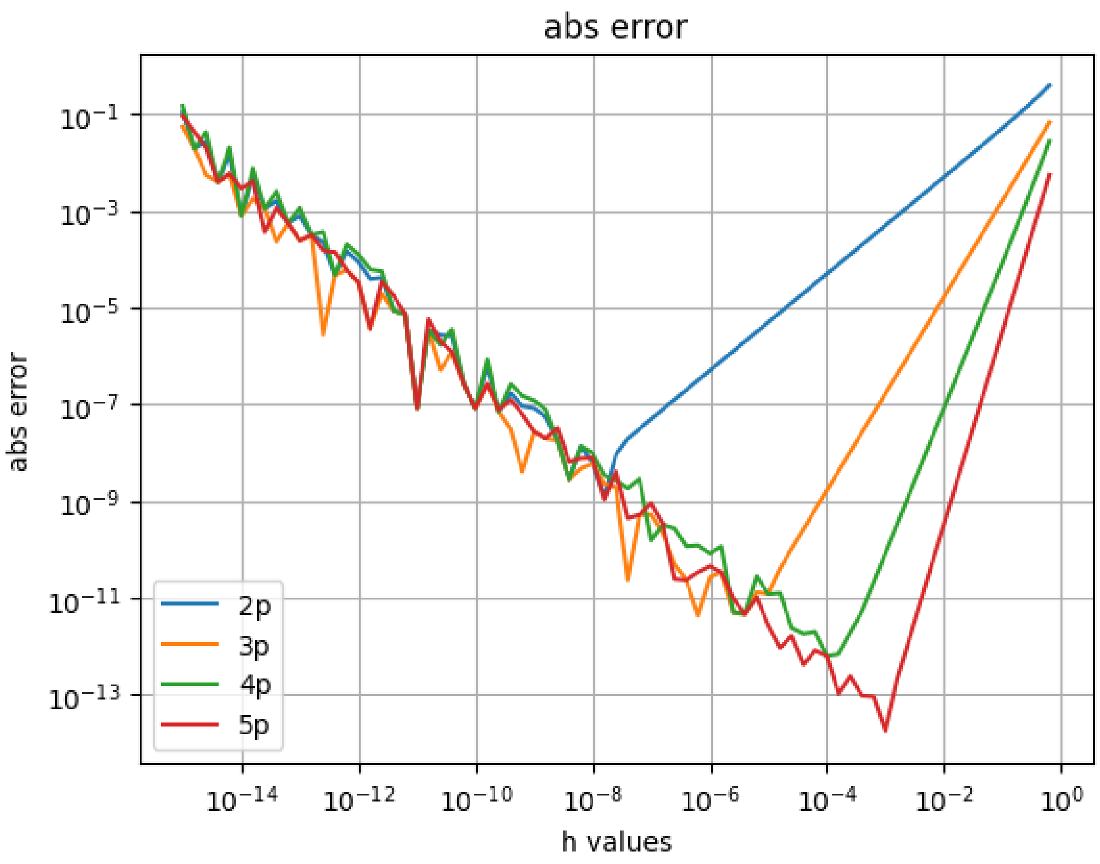
[3]:
```
x = 0
e_to_x = lambda x : math.e ** x
correct_diff = e_to_x(x)
i = range(1, 76)
h = [10 ** (-t / 5) for t in i]
direvatives = [calc_diff_2p(f = e_to_x, x = x, h = t) for t in h]
direvatives_3p = [calc_diff_3p(f = e_to_x, x = x, h = t) for t in h]
direvatives_4p = [calc_diff_4p(f = e_to_x, x = x, h = t) for t in h]
direvatives_5p = [calc_diff_5p(f = e_to_x, x = x, h = t) for t in h]
```

[4]:
```
# Draw everything

line_values = linspace(0, 70, 100)  # Generates 100 points between -10 and 10

# Create a list of y-values where each y-value is the constant
correct_values = [correct_diff] * len(line_values)

plot(correct_values)



plot(direvatives)
plot(direvatives_3p)
plot(direvatives_4p)
plot(direvatives_5p)
# plot(correct_diff, marker='s', linestyle='--', color='r', label='Line 2')
legend(['correct value', '2p', '3p', '4p', '5p'])

title('estimated direvative under standard scale')
xlabel('h values')
ylabel('Calculated Direvative')

# Add grid lines
grid(True)

# Show the plot
show()
```



```
[5]: errors_2p = [abs(direvative - 1) for direvative in direvatives]
     errors_3p = [abs(direvative - 1) for direvative in direvatives_3p]
     errors_4p = [abs(direvative - 1) for direvative in direvatives_4p]
     errors_5p = [abs(direvative - 1) for direvative in direvatives_5p]
```

```
[6]: loglog(h, errors_2p,label='2 p')
     loglog(h, errors_3p,label='3 p')
     loglog(h, errors_4p,label='4 p')
     loglog(h, errors_5p,label='5 p')

     title('abs error')
     xlabel('h values')
     ylabel('abs error')

     legend(['2p', '3p', '4p', '5p'])
     grid(True)
     show()
```



3 Exercise 3

From Exercise 1, we know Newton's interpolation polynomial with four points is

$$p_3(y) = f[x_1] + f[x_1x_2](y - x_1) + f[x, x_2x_3](y - x)(y - x_2) + f[x_1x_2x_3x_4](y - x_1)(y - x_2)(y - x_3)$$

If we take the third direvative of p_3, we got

$$\begin{aligned} p_3^{(3)} &= 3!f[x_1x_2x_3x_4] \\ &= 6f[x_1x_2x_3x_4] \\ &= 6 \cdot \frac{2f(x_4) - 6f(x_3) - 2f(x_2) + 6f(x_1)}{12h^3} \\ &= \frac{f(x_4) - 3f(x_3) - f(x_2) + 3f(x_1)}{h^3} \end{aligned}$$

,where

$$\begin{aligned}x_1 &= x\\ x_2 &= x-h\\ x_3 &= x+h\\ x_4 &= x+2h\end{aligned}$$

Thus

$$\begin{aligned}p_3^{(3)} &= 3!f[x_1x_2x_3x_4]\\ &= 6f[x_1x_2x_3x_4]\\ &= 6\cdot\frac{2f(x_4)-6f(x_3)-2f(x_2)+6f(x_1)}{12h^3}\\ &= \frac{f(x_4)-3f(x_3)-f(x_2)+3f(x_1)}{h^3}\\ &= \frac{f(x+2h)-3f(x+h)-f(x-h)+3f(x)}{h^3}\end{aligned}$$

Let's typlor expand $f(x+\frac{h}{2})$, we got

$$f(x+\frac{1}{2}h) = f(x)+\frac{1}{2}hf'(x)+\frac{h^2}{2^2 2!}f''(x)+\frac{h^3}{2^3 3!}f^{(3)}(x)+\frac{h^4}{2^4 4!}f^{(4)}(c)$$

where $c\in(x,x+\frac{h}{2})$

Let's also expand $f(x)$, $f(x+h)$, $f(x-h)$, $f(x+2h)$ about x, we got

$$\begin{aligned}f(x-h) &= f(x)-hf'(x)+\frac{h^2}{2!}f''(x)-\frac{h^3}{3!}f^{(3)}(x)+\frac{h^4}{4!}f^{(4)}(c_3)\\ f(x) &= f(x)\\ f(x+h) &= f(x)+hf'(x)+\frac{h^2}{2!}f''(x)+\frac{h^3}{3!}f^{(3)}(x)+\frac{h^4}{4!}f^{(4)}(c_2)\\ f(x+2h) &= f(x)+2hf'(x)+\frac{(2h)^2}{2!}f''(x)+\frac{(2h)^3}{3!}f^{(3)}(x)+\frac{(2h)^4}{4!}f^{(4)}(c_4)\end{aligned}$$

Thus,

$$f(x+2h)-3f(x+h)-f(x-h)+3f(x) = \frac{f^{(3)}(x)}{h^3}+h^4(\frac{2}{3}f^{(4)}(c_4)-\frac{1}{8}f^{(4)}(c_2)-\frac{1}{24}f^{(4)}(c_3))$$

where $c_4\in(x,2h)c_2\in(x,h), c_3\in(x-h,x)$, all of which converges to a constant as $h\to 0$

Thus,

$$f(x+2h)-3f(x+h)-f(x-h)+3f(x)=\frac{f^{(3)}(x)}{h^3}+C\cdot h^4$$

, where

$$C=\lim_{h\to 0}(\frac{2}{3}f^{(4)}(c_4)-\frac{1}{8}f^{(4)}(c_2)-\frac{1}{24}f^{(4)}(c_3))$$

Thus, according previous calculation, we have

$$f^{(3)}(x)=\frac{f(x+2h)-3f(x+h)-f(x-h)+3f(x)}{h^3}=f^{(3)}(x)+C\cdot h$$

, which means the error is $O(h)$

We do a tylor expansion around $(x+\frac{h}{2})$ and get:

$$f(x+2h)=f(x+\frac{h}{2})+\frac{3h}{2}f'(x+\frac{h}{2})+\frac{(\frac{3h}{2})^2}{2!}f''(x+\frac{h}{2})+\frac{(\frac{3h}{2})^3}{3!}f^{(3)}(x+\frac{h}{2})+\frac{(\frac{3h}{2})^4}{4!}f^{(4)}(x+\frac{h}{2})+\frac{(\frac{3h}{2})^5}{5!}f^{(5)}(a_1) \tag{1}$$

$$f(x+h)=f(x+\frac{h}{2})+\frac{h}{2}f'(x+\frac{h}{2})+\frac{(\frac{h}{2})^2}{2!}f''(x+\frac{h}{2})+\frac{(\frac{h}{2})^3}{3!}f^{(3)}(x+\frac{h}{2})+\frac{(\frac{h}{2})^4}{4!}f^{(4)}(x+\frac{h}{2})+\frac{(\frac{h}{2})^5}{5!}f^{(5)}(a_2) \tag{2}$$

$$f(x+\frac{1}{2}h)=f(x+\frac{1}{2}h) \tag{3}$$

$$f(x)=f(x+\frac{h}{2})-\frac{h}{2}f'(x+\frac{h}{2})+\frac{(\frac{h}{2})^2}{2!}f''(x+\frac{h}{2})-\frac{(\frac{h}{2})^3}{3!}f^{(3)}(x+\frac{h}{2})+\frac{(\frac{h}{2})^4}{4!}f^{(4)}(x+\frac{h}{2})-\frac{(\frac{h}{2})^5}{5!}f^{(5)}(a_3) \tag{4}$$

$$f(x-h)=f(x+\frac{h}{2})-\frac{3h}{2}f'(x+\frac{h}{2})+\frac{(\frac{3h}{2})^2}{2!}f''(x+\frac{h}{2})-\frac{(\frac{3h}{2})^3}{3!}f^{(3)}(x+\frac{h}{2})+\frac{(\frac{3h}{2})^4}{4!}f^{(4)}(x+\frac{h}{2})-\frac{(\frac{3h}{2})^5}{5!}f^{(5)}(a_4) \tag{5}$$

$$\tag{6}$$

, where $a_1\in(x+\frac{h}{2},x+2h), a_2\in(x+\frac{h}{2},x+h), a_3\in(x,x+\frac{h}{2}), a_4\in(x-h,x+\frac{h}{2})$

After doing the same calculation: $(f(x+2h)-3f(x+h)-f(x-h)+3f(x))$

We get

$$f(x+2h)-3f(x+h)-f(x-h)+3f(x) \tag{7}$$

$$=(\frac{9h^3}{2}-\frac{h^3}{2})f^{(3)}(x+\frac{h}{2})+h^5(\frac{(\frac{3}{2})^5}{5!}f^{(5)}(a_1)-3\frac{(\frac{1}{2})^5}{5!}f^{(5)}(a_2)-3\frac{(\frac{1}{2})^5}{5!}f^{(5)}(a_3)+\frac{(\frac{3h}{2})^5}{5!}f^{(5)}(a_4)) \tag{8}$$

Similarly, we got

$$f(x+2h) - 3f(x+h) - f(x-h) + 3f(x) = h^3 f^{(3)}(x + \frac{h}{2}) + A \cdot h^5 \tag{9}$$

, where

$$A = \lim_{(h\to 0)} (\frac{(\frac{3}{2})^5}{5!} f^{(5)}(a_1) - 3\frac{(\frac{1}{2})^5}{5!} f^{(5)}(a_2) - 3\frac{(\frac{1}{2})^5}{5!} f^{(5)}(a_3) + \frac{(\frac{3h}{2})^5}{5!} f^{(5)}(a_4)) \tag{10}$$

Thus,

$$f^{(3)}(x + \frac{h}{2}) = \frac{f(x+2h) - 3f(x+h) - f(x-h) + 3f(x)}{h^3} - A \cdot h^2 \tag{11}$$

Thus, the error is up to $O(h^2)$

3.1 Explanation

When we are evaluate the direvative at x, we are essentially using a not symmetric approximation. Yet when we evaluate the direvative as $x + \frac{h}{2}$, we make the system a symmetric system. The symetric approximation has a higher order error but this error is relative to $x + \frac{h}{2}$. Also, $f(x + \frac{h}{2})$ is cancelled out when we are doing our calculation, just like when we are evluating $f'(x)$ at x with $x - h, x, x + h$, we cancelled the $f(x)$ point

4 Exercise 4

[7]:
```python
def tp_integral(f, N: int, a: int, b:int):
    area = 0
    for i in range(0, N + 1): # N + 1 excluded
        x_i = a + i * ((b - a) / N)
        x_i_next = a + (i + 1) * ((b - a) / N)
        h = (x_i_next - x_i)
        curr_a = (h / 2) * (f(x_i) + f(x_i_next))
        area += curr_a
    return area
```

[8]:
```python
def mp_integral(f, N:int, a:int, b:int):
    area = 0
    for i in range(0, N + 1): # N + 1 excluded
        x_i = a + i * ((b - a) / N)
        x_i_next = a + (i + 1) * ((b - a) / N)
        h = (x_i_next - x_i)
        curr_a = (h) * (f((x_i + x_i_next) / 2))
        area += curr_a
    return area
```

[9]:
```python
f_sin = lambda x : math.sin(x)
```

```
[10]: N_range = range(1, 1001)
```

```
[11]: N_range
```

```
[11]: range(1, 1001)
```

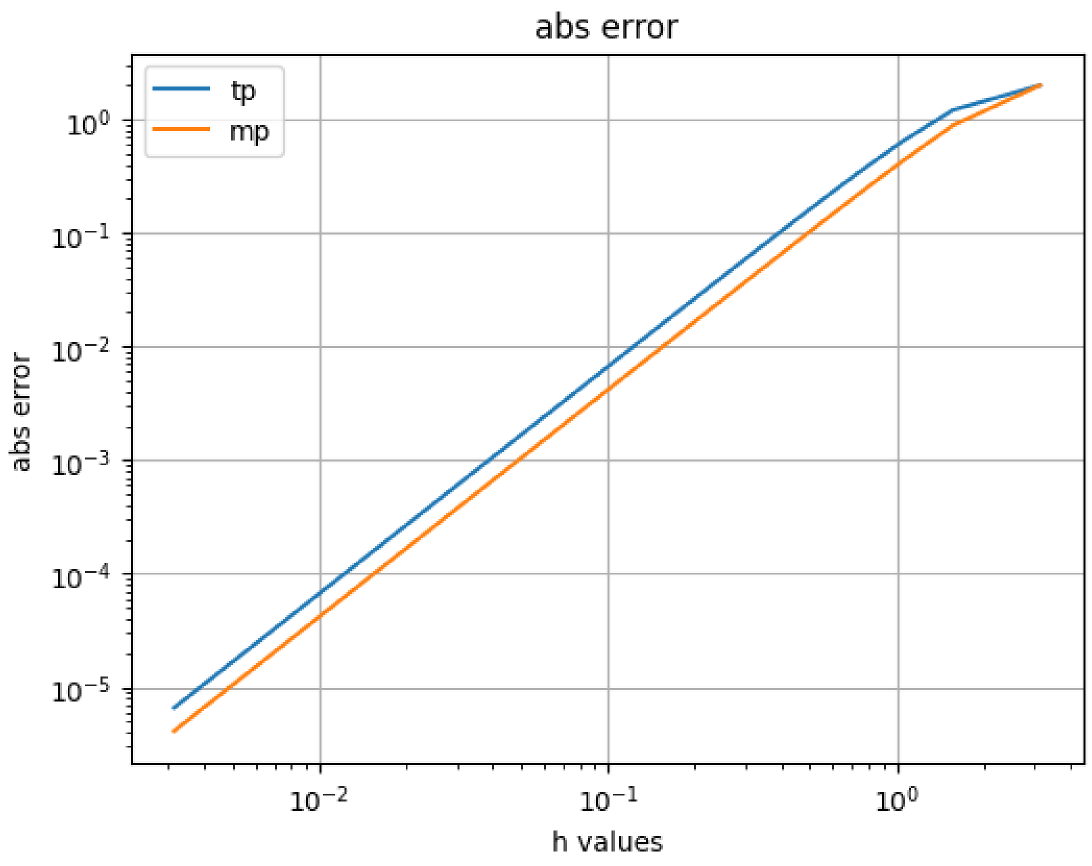
```
[12]: tp_result = [tp_integral(f_sin, i, 0, math.pi) for i in N_range]
      mp_result = [mp_integral(f_sin, i, 0, math.pi) for i in N_range]
      h_s = [math.pi / i for i in N_range]
```

```
[13]: error_tp = [abs(i - 2) for i in tp_result]
      error_mp = [abs(i - 2) for i in mp_result]
```

```
[14]: error_tp[:5], error_mp[:5]
```

```
[14]: ([2.0,
        1.2146018366025513,
        0.6396504768243363,
        0.3815612856978581,
        0.25089258495624045],
       [2.0,
        0.8892792654604085,
        0.4292036732051028,
        0.2482145589881113,
        0.16087962694644253])
```

```
[15]: loglog(h_s, error_tp,label='tp')
      loglog(h_s, error_mp,label='mp')

      title('abs error')
      xlabel('h values')
      ylabel('abs error')

      legend(['tp', 'mp'])
      grid(True)
      show()
```



[16]:
```
# hh_s = [i**2 for i in h_s]
# plot(hh_s, tp_result)
# plot(hh_s, mp_result)
# legend(['tp', 'mp'])
# grid(True)
# show()
```

4.1 Explanation for (1)

The midpoint method is a 1-point integration method because it approximates the area under the curve using rectangles. For each subinterval, the method takes the function's value at the midpoint and assumes that this value holds across the entire subinterval. This essentially creates a rectangle with height equal to the function's value at the midpoint and width equal to the subinterval's width. The area of this rectangle is taken as the approximation of the integral over that subinterval. The trapezoid method, on the other hand, uses trapezoids to approximate the area. For each subinterval, it takes the function's value at both endpoints, creating a trapezoid whose area is calculated using the average of these two values for the height and the subinterval's width for the base. This accounts for the change in the function's value over the interval, making it a 2-point method. Yet for both the rectangle and trapezoid are evaluating with the "same" area in a sense that the area of trapezoid under and above the rectangle is cancelled out. Thus, under a small interval, both method gives the same estimation.|

4.2 Explanation for (2)

Indeed when we are talking about the error in terms of the interval x_i, x_{i+1}. As shown in the formula, the error term is given by $\frac{h^3}{12} f^{''}(c)$, but when we are integral over $[a, b]$, we are summing over a lot of intervals, whose length is determined by h ($\Delta x = \frac{(b-a)}{h}$). Thus, actually when we are evaluating over interval $[a, b]$, we are doing a summation:

$$\begin{aligned}
\int_a^b f(x) &= \sum_{i=0}^{N-1} (\frac{h}{2} \Delta x_i - \frac{h^3}{12} f^{''}(c_i)) && (12) \\
&= \sum_{i=0}^{N-1} (\frac{h}{2} \Delta x_i) - \sum_{i=0}^{N-1} (\frac{h^3}{12} f^{''}(c_i)) && (13) \\
& && (14)
\end{aligned}$$

Where

$$\begin{aligned}
\Delta x_i &= x_{i+1} - x_i && (15) \\
x_i &= a + ih && (16) \\
h &= \frac{b-a}{N} && (17) \\
i &\in \{0, 1, 2, ..., N\} && (18) \\
c_i &\in [x_i, x_{i+1}] && (19) \\
& && (20)
\end{aligned}$$

When take a closer look on the term $-\sum_{i=0}^{N-1} (\frac{h^3}{12} f^{''}(c_i))$ To analyze the error, we take an absolute value, then

$$|\sum_{i=0}^{N-1} (\frac{h^3}{12} f^{''}(c_i))| \leq |\sum_{i=0}^{N-1} h^3 M_i| \leq |\sum_{i=0}^{N-1} h^3 M| \leq |M| |\sum_{i=0}^{N-1} h^3| = M |\sum_{i=0}^{N-1} h^3| = M \cdot N \cdot (\frac{b-a}{N})^3 = \frac{(b-a)^3}{N^2} = (b-a) \quad (21)$$

where \$ i, |f^{' '}(c_i)| <M_i <M \$.

Thus, we figured that the overall error term should be of the order $O(h^2)$

5 Exercise 5

[17]:
```
coe_dict = {
    2: [
        [-math.sqrt(1 / 3), 1.0],
        [math.sqrt(1 / 3), 1.0]
    ],
    3: [
        [-math.sqrt(3 / 5), 5/9],
        [0.0, 8/9],
        [math.sqrt(3 / 5), 5/9]
    ],
    4: [
        [(-(math.sqrt((15 + 2 * math.sqrt(30)) / 35))), (90 - 5* sqrt(30)) /
 ↪180],
        [(-(math.sqrt((15 - 2 * math.sqrt(30)) / 35))), (90 + 5* sqrt(30)) /
 ↪180],
        [((math.sqrt((15 - 2 * math.sqrt(30)) / 35))),  (90 + 5* sqrt(30)) /
 ↪180],
        [((math.sqrt((15 + 2 * math.sqrt(30)) / 35))),  (90 - 5* sqrt(30)) /
 ↪180]
    ]
}
```

[18]:
```
coe_dict[4]
```

[18]:
```
[[-0.8611363115940526, 0.34785484513745385],
 [-0.33998104358485626, 0.6521451548625462],
 [0.33998104358485626, 0.6521451548625462],
 [0.8611363115940526, 0.34785484513745385]]
```

[19]:
```
def gq_integral_unscaled(f, n: int):
    para_list = coe_dict[n]
    s = 0.0
    for i in para_list:
        # print(f"selected parameter: {para_list}, n: {n}")
        curr_s = f(i[0]) * i[1]
        s += curr_s

    return s
```

[20]:
```
gq_integral_unscaled(math.sin, 2) # passed
```

[20]:
```
0.0
```

[21]:
```
gq_integral_unscaled(math.cos, 4)
```

[21]:
```
1.6829416886959736
```

Passed basic unit test cases

[22]:
```
f_1 = lambda x: 1+2*x
f_2 = lambda x: f_1(x) + 3*(x**2)
f_3 = lambda x: f_2(x) + 4*(x**3)
```

```
f_4 = lambda x: f_3(x) + 5*(x**4)
f_5 = lambda x: f_4(x) + 6*(x**5)
f_6 = lambda x: f_5(x) + 7*(x**6)
f_7 = lambda x: f_6(x) + 8*(x**7)
f_list = [f_1, f_2, f_3, f_4, f_5, f_6, f_7]
```

[23]:
```
test_r = [gq_integral_unscaled(each_f, 4) for each_f in f_list]
```

[24]:
```
test_r
```

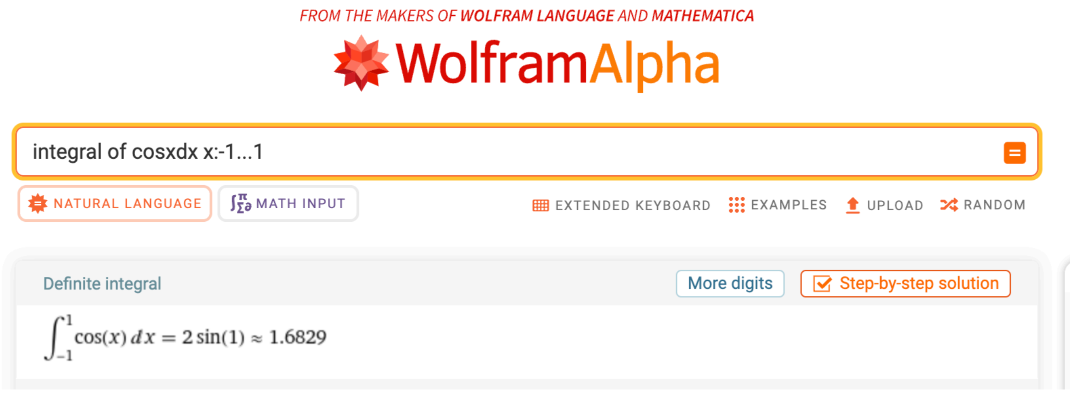
[24]:
```
[2.0, 4.0, 3.9999999999999996, 6.0, 6.0, 7.999999999999999, 7.999999999999999]
```

[25]:
```
def gq_integral_scaled(f, n: int, a: float, b: float):
    #          f_n = lambda x: ((x1-x0)/2) * f(((x1-x0)/2)*x +(x1+x0)/2)
    f_scaled = lambda t: ((b - a) / 2) * f(((b - a) / 2) * t + ((a + b) / 2)) #␣
 ↪t: from -1 to 1 now
    # f_n = lambda x: ((b-a)/2) * f(((b-a)/2)*x +(b+a)/2)
    return gq_integral_unscaled(f = f_scaled, n = n)
```

[26]:
```
gq_integral_scaled(math.cos, 4, -1, 1) # passed basic case
```

[26]:
```
1.6829416886959736
```

The picture shows that the answer is correct


FROM THE MAKERS OF WOLFRAM LANGUAGE AND MATHEMATICA

WolframAlpha

integral of cosxdx x:-1...1

NATURAL LANGUAGE | MATH INPUT | EXTENDED KEYBOARD | EXAMPLES | UPLOAD | RANDOM

Definite integral | More digits | Step-by-step solution

$\int_{-1}^{1} \cos(x)\, dx = 2\sin(1) \approx 1.6829$

[27]:
```
test_r_2 = [gq_integral_scaled(each_f, 4, -1, 1) for each_f in f_list]
```

[28]:
```
test_r_2
```

[28]:
```
[2.0, 4.0, 3.9999999999999996, 6.0, 6.0, 7.999999999999999, 7.999999999999999]
```

[29]:
```
gq_integral_scaled(math.sin, 4, 0, math.pi) # expected
```

[29]:
```
1.9999842284577225
```

```
[30]: def gq_integral(f, n: int, a: float, b: float, N: int):
          integ = 0.0
          for i in range(0, N): # This is the final problem
              x_i      = float(a) +  (float(i)*     float((b - a) / N))
              x_i_next = float(a) + (float(i + 1)*float((b - a) / N))
              integ += float(gq_integral_scaled(f = f,
                                                n = n,
                                                a = x_i,
                                                b = x_i_next))
          return integ
```

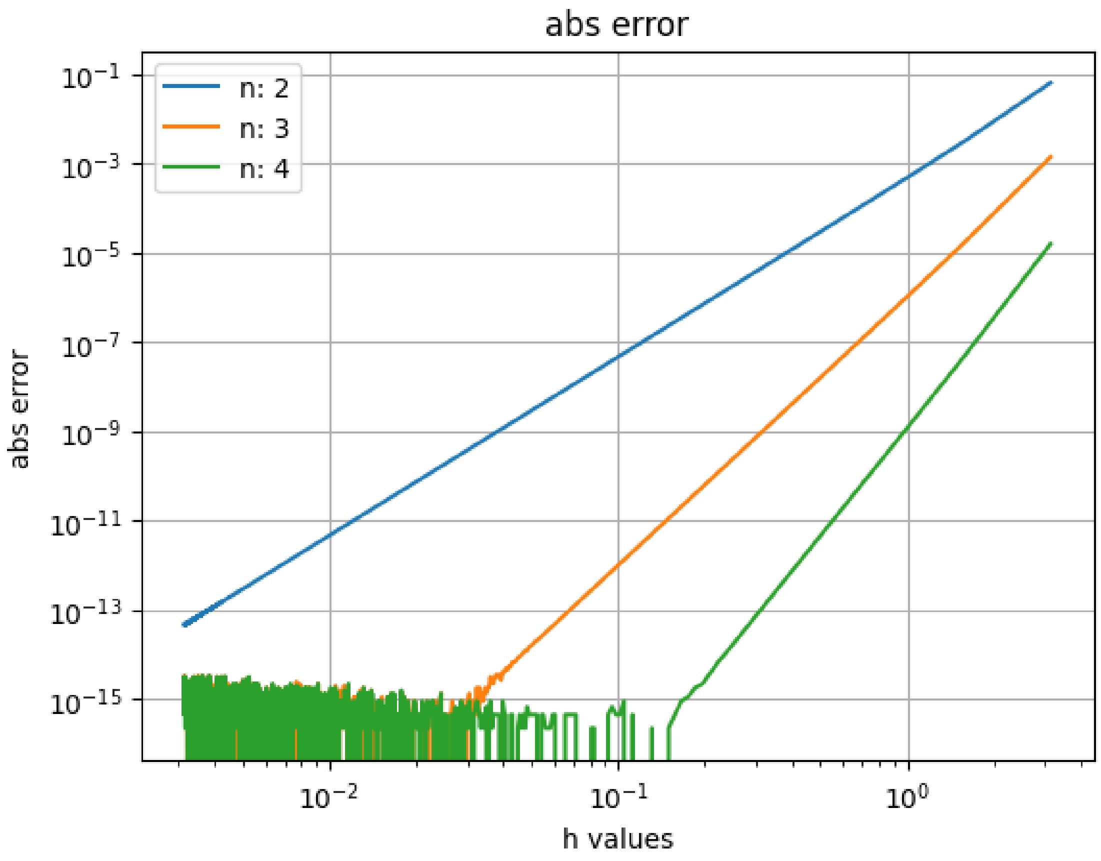
```
[31]: gq_integral(math.sin, 3, -math.pi, math.pi, N=1000)
```

```
[31]: -5.196885944584034e-16
```

```
[32]: def test_and_draw_graph(f, a: float, b: float, N_range:int, correct_answer:
       ↪float):
          N_range_G_list = range(1, N_range + 1)
          n_2_test_result = [gq_integral(f, 2, float(a), float(b), N = i) for i in␣
       ↪N_range_G_list]
          n_3_test_result = [gq_integral(f, 3, float(a), float(b), N = i) for i in␣
       ↪N_range_G_list]
          n_4_test_result = [gq_integral(f, 4, float(a), float(b), N = i) for i in␣
       ↪N_range_G_list]

          n_2_error = [fabs(correct_answer - j) for j in n_2_test_result]
          n_3_error = [fabs(correct_answer - j) for j in n_3_test_result]
          n_4_error = [fabs(correct_answer - j) for j in n_4_test_result]

          h_intevals = [((b - a) / i) for i in N_range_G_list]

          loglog(h_intevals, n_2_error)
          loglog(h_intevals, n_3_error)
          loglog(h_intevals, n_4_error)

          title('abs error')
          xlabel('h values')
          ylabel('abs error')

          legend(['n: 2', 'n: 3', 'n: 4'])
          grid(True)
          show()
          return n_4_test_result
```

```
[33]: n_4_test_result = test_and_draw_graph(math.sin, a=0, b = math.pi, N_range =␣
       ↪1000, correct_answer = 2.0)
```

```
# From the graph below we can see from the slop of the line that they are␣
↪producing the correct result
```



```
[ ]:
```

```
[ ]:
```