**EX.5.1.3, Sauer3 – Problem rewritten by Julien.**
We want to study the two-point forward-difference formula to approximate $f'(x)$ where $f(x) = \sin x$ and $x = \frac{\pi}{3}$.

a. (handwritten) For this problem, what is $f'(x)$?

b. (handwritten) What is the two-point forward-difference formula in its general form?

c. (python) Use the two-point forward-difference formula to approximate $f'(x)$ for (a) $h = 0.1$, (b) $h = 0.01$, and (c) $h = 0.001$? Observe that, as $h$ gets smaller, the approximation gets better. (However if we were to take $h$ too small, we would suffer from cancellation error and start losing accuracy.)

d. (handwritten) What is the two-point forward-difference approximation error in its general form? (Hint: There is a "$c$" in the formula. Specify an interval where $c$ is.)

e. (handwritten) Since we work with small positive $h$'s, let us assume that $h$ is in $(0, 0.5)$, so that $x + h$ is in $(\frac{\pi}{3}, \frac{\pi}{2})$, find lower and upper bounds for the two-point forward-difference approximation error (independent of $c$). We call this an "interval where the approximation error is". We want this interval to be "relevant".

f. (python) Check that the "true approximation error" is within your interval of part (e).

g. (handwritten) In general, we are more interested in guaranteeing that "the error is not larger than". We mean the absolute value of the error. (As opposed to find an interval where the error is.) Based on part (e), find an upper bound for the absolute value of the "true approximation error". We call this an "upper bound on the approximation error".

h. (python) Check that an "upper bound on the approximation error" is an upper bound on the approximation error. We want this upper bound to be "relevant".



**EX.5.1.3, Sauer3, solution, Langou**

Colab: `https://colab.research.google.com/drive/1rcWecA_rDytR9eCWrSuf2GpHAi0-Gt5P`

(a) (handwritten) For this problem, what is $f'(x)$?

$$f'(x) = 0.5.$$

Note: For this problem, we know the value of which we want to compute an approximation. We want to compute an approximation of $f'(x)$, where $f(x) = \sin x$ and $x = \frac{\pi}{3}$. Since the derivative of sin is cos and $\cos(\frac{\pi}{3})$ is 0.5. We are trying to compute an approximation of 0.5. (The closer the approximation is from 0.5, the better the approximation.)

(b) (handwritten) What is the two-point forward-difference formula in its general form?

Given an $f$, an $x$, and an $h$, the two-point forward-difference formula is

$$\frac{1}{h}\left(f(x+h) - f(x)\right).$$

For small $h$, it approximates $f'(x)$.

Note: $h$ needs to be positive. If $h$ is zero, the formula is not defined. If $h$ is negative, this becomes the "two-point backward-difference formula".

(c) (python) Use the two-point forward-difference formula to approximate $f'(x)$ for (a) $h = 0.1$, (b) $h = 0.01$, and (c) $h = 0.001$? Observe that, as $h$ gets smaller, the approximation gets better. (However if we were to take $h$ too small, we would suffer from cancellation error and start losing accuracy.)

We use the formula to approximate $f'(x)$, where $f(x) = \sin x$ and $x = \frac{\pi}{3}$, and then (a) $h = 0.1$, (b) $h = 0.01$, and (c) $h = 0.001$, and we get

| h | approximation |
|---|---|
| 0.1 | 0.455902 |
| 0.01 | 0.495662 |
| 0.001 | 0.499567 |

Note: We know that $f'(x)$ is 0.5. So we are trying to compute 0.5. We see from the table that the smaller the $h$, the better the accuracy. (However if we were to take $h$ too small, we would suffer from cancellation error and start losing accuracy.)

```
from math import sin
from math import cos
from math import pi

f  = lambda x : sin(x)
fp = lambda x : cos(x)

two_point_forward_difference_formula = lambda x, h : ( f(x+h) - f(x) ) / h

x = pi / 3

for h in [ 0.1, 0.01, 0.001 ]:

  print(  f"{h:5.0e}",
          f"{two_point_forward_difference_formula( x, h ):9.6f}")
```

```
1e-01  0.455902
1e-02  0.495662
1e-03  0.499567
```

(d) (handwritten) What is the two-point forward-difference approximation error in its general form? (Hint: There is a "$c$" in the formula. Specify an interval where $c$ is.)

The two-point forward-difference formula and its error is given in the book with Equation (5.3):

$$f'(x) = \frac{1}{h}\left(f(x+h) - f(x)\right) - \frac{h}{2}f''(c), \qquad \text{where } c \text{ is between } x \text{ and } x+h.$$

Note: In other words, the equation above reads: (what we want to compute, $f'(x)$) is equal to (the formula used to approximate it, $\frac{1}{h}\left(f(x+h) - f(x)\right)$) plus (the error of the formula, $-\frac{h}{2}f''(c)$).

Note: We cannot know $c$ but we can try to use the knowledge that $c$ is between $x$ and $x+h$ to get some useful information, such as bounds on the error.

Note: If we were to know $c$, we simply would compute $\left(\text{the error of the formula}, -\frac{h}{2}f"(c)\right)$ and add it to $\left(\text{the formula}, \frac{1}{h}\left(f(x+h)-f(x)\right)\right)$ to get our (what we want to compute, $f'(x)$) exactly! In practice, computing $c$ is really hard, not practical.

Note: We call $\left(f'(x)-\frac{1}{h}\left(f(x+h)-f(x)\right)\right)$ the "approximation error" or sometimes the "true approximation error".

(e) (handwritten) Since we work with small positive $h$'s, let us assume that $h$ is in $(0, 0.5)$, so that $x+h$ is in $(\frac{\pi}{3}, \frac{\pi}{2})$, find lower and upper bounds for the two-point forward-difference approximation error (independent of $c$). We call this an "interval where the approximation error is". We want this interval to be "relevant".

Because $f$ is sin, we know that

$$f"(c) = -\sin(c).$$

Because $c$ is in $(x, x+h)$, $c$ is in $(\frac{\pi}{3}, \frac{\pi}{2})$, so that

$$-1 \leq f"(c) \leq -\frac{\sqrt{3}}{2}.$$

So that

$$\frac{\sqrt{3}}{4}h \leq -\frac{h}{2}f"(c) \leq \frac{1}{2}h.$$

And so an "interval where the approximation error is" is

$$\frac{\sqrt{3}}{4}h \leq \left(f'(x)-\frac{1}{h}\left(f(x+h)-f(x)\right)\right) \leq \frac{1}{2}h.$$

And so an "interval where the approximation error is" is

$$(\frac{\sqrt{3}}{4}h, \frac{1}{2}h).$$

Note: We will say that this interval is relevant because (1) as $h$ goes to zero, the points in the interval go to zero too; (2) the interval lower and upper bounds do not need $c$, only on $h$ and $x$ and $f$.

Note: An interval that is not relevant for example would be something like $(0, 1)$. It is true that the approximation error is in this interval but the interval $(0, 1)$ is too big to give any useful information.

Note: Also, there is no $c$ in our interval. And this is important. In practical scenarios, we do not know $c$, so we cannot compute the "true approximation error". However we can compute the "interval where the approximation error is". Provided that there is no $c$. (Only $x$ and $h$ is allowed.)

Note: While we do not know the "true approximation error", note that the interval is very tight and gives a lots of insights. See the next question for some computation. In general for an error, we do not need to know the error with much accuracy (actually as already explained, we cannot in general compute the error with much accuracy), we just want to know: "is the error $10^{-1}$ or $10^{-4}$ or $10^{-12}$?"

(f) (python) Check that the "true approximation error" is within your interval of part (e).

We use our first interval formula:

$$(\frac{\sqrt{3}}{4}h, \frac{1}{2}h).$$

We get

| h | approximation | approximation error | interval where approximation error is |
|---|---|---|---|
| 0.1 | 0.455902 | $4.410 \cdot 10^{-2}$ | $(4.330 \cdot 10^{-2}, 5.000 \cdot 10^{-2})$ |
| 0.01 | 0.495662 | $4.338 \cdot 10^{-3}$ | $(4.330 \cdot 10^{-3}, 5.000 \cdot 10^{-3})$ |
| 0.001 | 0.499567 | $4.331 \cdot 10^{-4}$ | $(4.330 \cdot 10^{-4}, 5.000 \cdot 10^{-4})$ |

Note: We should check that

a. for $h = 0.1$, indeed, the approximaton error $(4.410 \cdot 10^{-2})$ is in the interval $(4.330 \cdot 10^{-2}, 5.000 \cdot 10^{-2})$.

b. for $h = 0.01$, indeed, the approximaton error $(4.338 \cdot 10^{-3})$ is in the interval $(4.330 \cdot 10^{-3}, 5.000 \cdot 10^{-3})$.

c. for $h = 0.001$, indeed, the approximaton error $(4.331 \cdot 10^{-4})$ is in the interval $(4.330 \cdot 10^{-4}, 5.000 \cdot 10^{-4})$.

```
from math import sin, cos, pi, sqrt

f  = lambda x : sin(x)

two_point_forward_difference_formula = lambda x, h : \
  ( f(x+h) - f(x) ) / h

# remember that, in practical scenario, we will not know fp(x),
# since fp(x) is what we are trying to approximate,
# so, in practical scenario, we cannot compute 'approximation_error'
# in this 'toy' scenario, we know fp(x),
# so we can compute 'approximation_error'
fp = lambda x : cos(x)
approximation_error = lambda x, h : \
  fp(x) - two_point_forward_difference_formula( x, h )

# these lower and upper bounds are only valid at x = pi/3,
# so we postfix _atx60 to make this clear
# see handwritten part to see the derivation of these lower bounds
lower_bound_atx60 = lambda h : \
  sqrt(3) / 4. * h

upper_bound_atx60 = lambda h : \
  h / 2.

x = pi / 3.

for h in [ 0.1, 0.01, 0.001 ]:

  print(  f"{h:5.0e}",
          f"{two_point_forward_difference_formula( x, h ):9.6f}",
          " | ",
          f"{lower_bound_atx60( h ):6.3e}",
          f"{approximation_error( x, h ):6.3e}",
          f"{upper_bound_atx60( h ):6.3e}")
```

```
1e-01  0.455902  |  4.330e-02 4.410e-02 5.000e-02
1e-02  0.495662  |  4.330e-03 4.338e-03 5.000e-03
1e-03  0.499567  |  4.330e-04 4.331e-04 5.000e-04
```

(g) (handwritten) In general, we are more interested in guaranteeing that "the error is not larger than". We mean the absolute value of the error. (As opposed to find an interval where the error is.) Based on part (e), find an upper bound for the absolute value of the "true approximation error". We call this an "upper bound on the approximation error".

In part (e), we obtained the following interval:

$$(\frac{\sqrt{3}}{4}h, \frac{1}{2}h).$$

for the approximation error. Therefore, we can bound the absolute value of the approximation error by $\frac{1}{2}h$. We get

$$\left| f'(x) - \left( \frac{1}{h} \left( f(x+h) - f(x) \right) \right) \right| \leq \frac{1}{2}h.$$

Note: We only do questions (g) and (h) for our first interval formula: $(\frac{\sqrt{3}}{4}h, \frac{1}{2}h)$. A similar reasoning for our second interval formula.

(h) (python) Check that an "upper bound on the approximation error" is an upper bound on the approximation error. We want this upper bound to be "relevant".

We get

| h | approximation | absolute value of approximation error | upper bound |
|---|---|---|---|
| 0.1 | 0.455902 | $4.410 \cdot 10^{-2}$ | $5.000 \cdot 10^{-2}$ |
| 0.01 | 0.495662 | $4.338 \cdot 10^{-3}$ | $5.000 \cdot 10^{-3}$ |
| 0.001 | 0.499567 | $4.331 \cdot 10^{-4}$ | $5.000 \cdot 10^{-4}$ |

Note: We should check that

a. for $h = 0.1$, indeed, the absolute value of the approximaton error $(4.410 \cdot 10^{-2})$ is less than $5.000 \cdot 10^{-2}$.

b. for $h = 0.01$, indeed, the absolute value of the approximaton error $(4.338 \cdot 10^{-3})$ is less than $5.000 \cdot 10^{-3}$.

c. for $h = 0.001$, indeed, the absolute value of the approximaton error $(4.331 \cdot 10^{-4})$ is less than $5.000 \cdot 10^{-4}$.

```
from math import sin, cos, pi, sqrt

f  = lambda x : sin(x)

two_point_forward_difference_formula = lambda x, h : \
  ( f(x+h) - f(x) ) / h

# remember that, in practical scenario, we will not know fp(x),
# since fp(x) is what we are trying to approximate,
# so, in practical scenario, we cannot compute 'approximation_error'
# in this 'toy' scenario, we know fp(x), so we can compute 'approximation_error'
fp = lambda x : cos(x)
approximation_error = lambda x, h : \
  fp(x) - two_point_forward_difference_formula( x, h )

# this upper bound on abs of error is only valid at x = pi /3,
# so we postfix _atx60 to make this clear
# see handwritten part to see the derivation of these lower bounds
upper_bound_atx60 = lambda h : \
  h / 2.
```

```
x = pi / 3.

for h in [ 0.1, 0.01, 0.001 ]:

  print(  f"{h:5.0e}",
          f"{two_point_forward_difference_formula( x, h ):9.6f}",
          " | ",
          f"{abs(approximation_error( x, h )):6.3e}",
          f"{upper_bound_atx60( h ):6.3e}")
```

```
1e-01  0.455902  |  4.410e-02 5.000e-02
1e-02  0.495662  |  4.338e-03 5.000e-03
1e-03  0.499567  |  4.331e-04 5.000e-04
```