MASD assignment 2

Anton (ckf216), Simon (wgh762) og Kirstine (zbf480)

September 2020

Exercise 2

a)

i) This one would not give $\frac{df}{dt}$, since the value of Δt is not approaching 0. The expression is basically the expression of differentiating a function:

$$lim_{h\to 0}\frac{f(a+h)-f(a)}{h}$$

except without limiting it, as well as a being replaced with t, and h being replaced with Δt. Since limiting is an important part of the above expression, (i) as given in the assignment text would not be able to give $\frac{df}{dt}$ at time t

ii) This one has a simpler explanation:
It is able to give $\frac{df}{dt}$ at time t, this is because when approaching 0, it doesn't matter what constant is manipulating h, h will still approach 0, and as such, the constant can be ignored. This leaves us with the normal expression for differentiation. Which means that it is able to give $\frac{df}{dt}$ at time t

iii) This is even simpler, it follows the expression for differentiation to a tee:

$$lim_{h\to 0}\frac{f(a+h)-f(a)}{h}$$

with only the values being given new names ($a \to t, h \to \Delta t$). As such this would also be able to give $\frac{df}{dt}$ at time t.

b)

Excuse the less professional look of the second graph, a way to input the function into a graphing tool was not found.

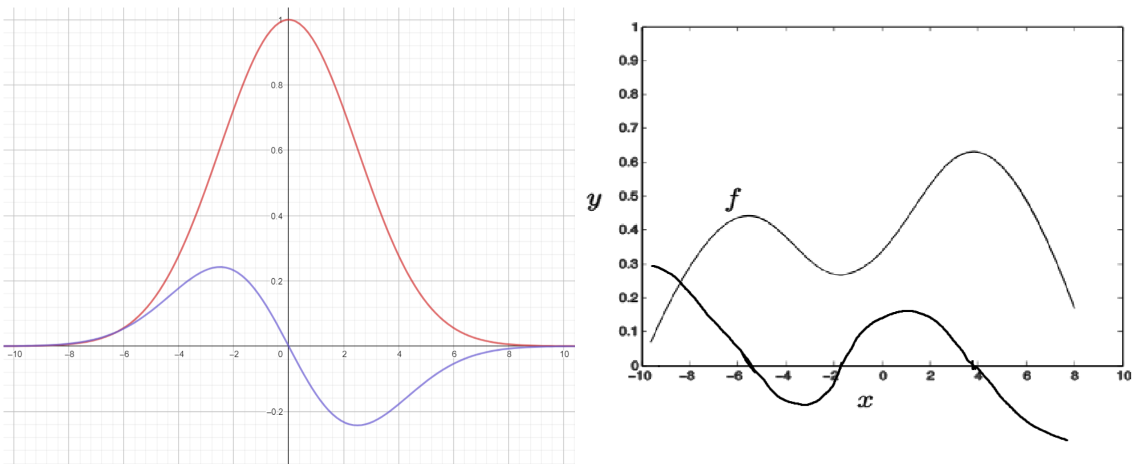

Figure 1: The graphs and their derivatives

c)

While the it is difficult to see, it can still be seen with a bit of zoom, that $x = 0$ is a local minima.

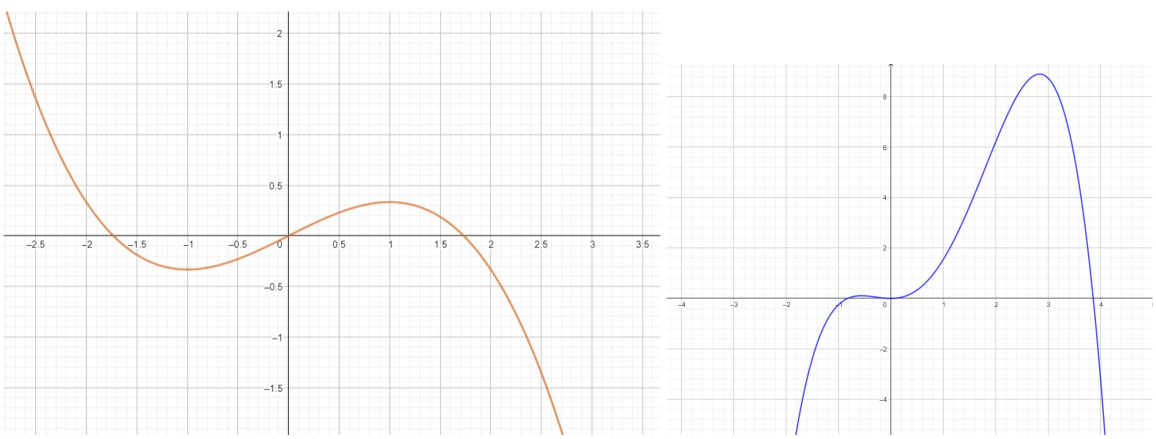

Figure 2: The graphs whose derivatives follow the assignment text

Exercise 3

a) $\frac{d}{dx}$ $(x^3 + e^{2x})$
first step is to apply power rule to x^3 and that gives us $3x^2$
next we apply chain rule to (e^{2x}) that gives us $e^{2x}\frac{d}{dx}(2x)$
then $\frac{d}{dx}(2x) = 2$ so the following result is $3x^2 + e^{2x} \cdot 2$

b) $\frac{d}{dx}$ $(e^{x^2+3x^3})$
we use chain rule: $\frac{df(u)}{dx} = \frac{df}{du} \cdot \frac{du}{dx}$
derivative of outer function is e, evaluated of inner function is $x^2 + 3x^3$.
Derivative of inner function is: $2x + 9x^2$
so the following result is $e^{x^2+3x^3}\left(2x + 9x^2\right)$

c) $\frac{d}{dx}$ $(\frac{ln(x)}{x2})$

We use the quotient rule with ln(x) being the first function and x^2 being the second. When calculating the derivative of x^2, we use the power rule.

$\frac{d}{dx} = \frac{(x^2)(\frac{1}{x})-ln(x)(2x)}{x^{2^2}}$

$\frac{d}{dx} = \frac{x-2xln(x)}{x^4}$

$\frac{d}{dx} = \frac{x(1-2ln(x))}{x(x^3)}$

$\frac{d}{dx} = \frac{1-2ln(x)}{x^3}$

d) $\frac{d}{dx}\left(e^{x^2+3xy+2y^3}\right)$
we use the chain rule: $e^{x^2+3xy+2y^3}\frac{d}{dx}\left(x^2 + 3xy + 2y^3\right)$
so in this case we multiply the original expression with the derivative of the inner function, with respect to x.
we end up with following result: $e^{x^2+3xy+2y^3}(2x + 3y)$

e) $\frac{\partial}{\partial y} = (e^{xy})(ln(x^2 + y^3))$

To find the partial derivative with respect to y, we first use the product rule. To take the derivative og $ln(x^2 + y^3)$, we use the chain rule, taking the derivative of ln and then of $(x^2 + y^3)$, where we need to use the sum rule, constant rule (as x is treated as a constant) and the power rule. To take the derivative of e^{xy}, we also have to use the chain rule and the constant multiple rule.

$\frac{\partial}{\partial y} = (e^{xy})(\frac{1}{x^2+y^3})(3y^2) + ln(x^2 + y^3)(e^{xy})(x)$

$\frac{\partial}{\partial y} = (\frac{3y^2e^{xy}}{x^2+y^3}) + xe^{xy}ln(x^2 + y^3)$

f) $\frac{\partial}{\partial x_i}(x^T Ax)$ where $x^T = \begin{pmatrix} x_1 \dots x_n \end{pmatrix}$ and $A = \begin{pmatrix} a_{11} \dots a_{1n} & \\ \vdots \ddots & \\ a_{n1} \dots a_{nn} & \end{pmatrix}$ $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$

first we calculate (Ax): $\begin{pmatrix} a_{11} \dots a_{1n} & \\ \vdots \ddots & \\ a_{n1} \dots a_{nn} & \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{bmatrix} a_{11}x_1 + ... + a_{1n}x_n \\ a_{n1}x_1 + ... + a_{nn}x_n \end{bmatrix}$

next we calculate $(Ax) \cdot x^T = \begin{bmatrix} a_{11}x_1 + ... + a_{1n}x_n \\ a_{n1}x_1 + ... + a_{nn}x_n \end{bmatrix} \cdot \begin{pmatrix} x_1 \dots x_n \end{pmatrix} = \begin{bmatrix} a_{11}x_1^2 + ... + a_{1n}x_n x_1 \\ a_{n1}x_1 x_n + ... + a_{nn}x_n^2 \end{bmatrix}$

now we will take the derivative with respect to every x: $\begin{pmatrix} \frac{d}{x_1} \\ \vdots \\ \frac{d}{x_n} \end{pmatrix}$

$$= \begin{pmatrix} 2a_{11}x_1 + ... + a_{1n}x_n \\ \vdots \\ a_{n1}x_1 + ... + 2a_{nn}x_n \end{pmatrix} = 2 \begin{pmatrix} a_{11}x_1 + ... + a_{1n}x_n \\ \vdots \\ a_{n1}x_1 + ... + a_{nn}x_n \end{pmatrix} = 2 \begin{pmatrix} a_{11} + ... + a_{1n} \\ \vdots \\ a_{n1} + ... + a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

so now we can evaluate our result, and quickly realize that we just ended up with 2 times our original matrix times original vector. So we can write the result as: $2A \cdot x$

g) first we make sure that the dimensions are correct:
$x \in \mathbb{R}^{nx1}, A \in \mathbb{R}^{nxn}$ then
we can use the differential rule of vectors: $\frac{d(x^T a)}{dx} = \frac{d(a^T x)}{dx} = a^T$ where $x, a \in \mathbb{R}^{nx1}$
now we can use the chain rule where we first treat Ax as constant, and then $x^T A$ to get following:
$\frac{d(x^T Ax)}{dx} = x^T A^T + x^T A$ where A $\in \mathbb{R}^{nxn}$
So after reducing we end up with: $\frac{d(x^T Ax)}{dx} = x^T(A^T + A)$

Exercise 4

a)

python code:

```
def badness(a, b, cho, nob):
    # declare list.
    numbersList = []
    # for loop causing the function to run code 20 times.
    for i in range(19):
        # putting every result in the list.
        firstNumber = (nob[i]-a*cho[i]-b)**2
```

```
        secondNumber = (nob[i+1]-a*cho[i+1]-b)**2
        puttingTogether = firstNumber + secondNumber
        numbersList.append(puttingTogether)
    # calculate the sum of every number.
    numbersList.append(((nob[18]-a*cho[18]-b)**2) + ((nob[19]-a*cho[19]-b)**2))
    theSum = sum(numbersList)
    result = (1/20*theSum)
    return result
```

b)

python code:

```
import torch
def badnessgradient(a, b, cho, nob):
    # declare list.
    numbersList = []
    # for loop causing the function to run code 20 times.
    for i in range(19):
        # calculating the gradient for a and b.
        x = torch.tensor(a, requires_grad=True)
        y = torch.tensor(b, requires_grad=True)
        z = (nob[i]-x*cho[i]-y)**2
        z = z.mean()
        z.backward()
        firstGradient = x.grad
        secondGradient = y.grad
        # adding a gradient and b gradient, and putting in list
        addition = firstGradient + secondGradient
        numbersList.append(addition)
    # take the sum of all 20 gradients
    result = sum(numbersList)
    return result
```