Assignment 2

MASD 2018
Department of Computer Science
University of Copenhagen

Casper Lisager Frandsen <fsn483@alumni.ku.dk>

Version 1
Due: 16. September 2018, 10:00

Contents

Exercise 1

(a)

This part of the assignment has been completed under the Peer Feedback system on Absalon.

Exercise 2

(a)

(a).1

This expression **will not** give $\frac{df}{dt}$ at time t, because Δt does not approach 0.

(a).2

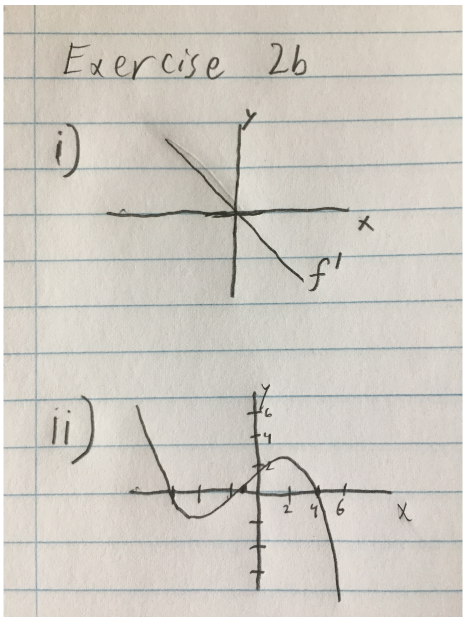
This expresssion **will** give $\frac{df}{dt}$ at time t. Since h is a constant, it does not matter whether we multiply anything to it, as long as we do the same to all instances of h

(a).3

This expresssion **will** give $\frac{df}{dt}$ at time t. Following the same logic as the previous expression, this expression is simply the same as multiplying h by a negative number. As told in the lectures, the formula should work for both positive and negative values of h. In this case, h is simply replaced by another name, Δt.

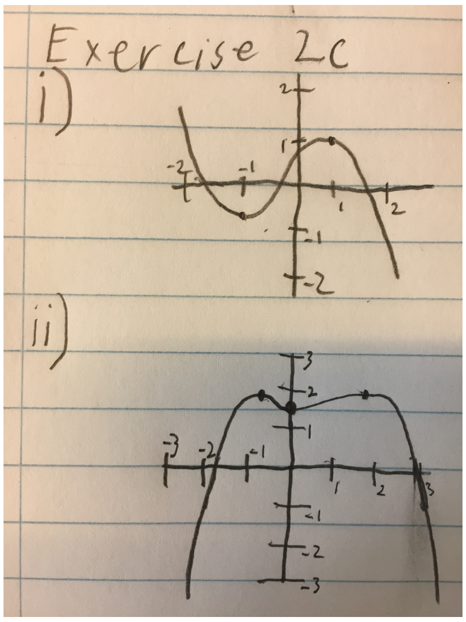
(b)

The relevant points for the given functions are the following:
Graph 1:
The function $f'(x)$ intersects the x-axis in 0. $f'(0) = 0$
Graph 2:
The relevant points of function $f'(x)$ are the following:
The function intersects the x-axis in all points where $f(x)$ has local maxima and minima.



(c)

The relevant points for the given functions are as follows:
Graph 1:
$f(x)$ has local maxima and minima in $x = (-1)$ and $x = 1$.
$f(x)$ decreases when approaching $f(-1)$, increases between $f(-1)$ and $f(1)$, and again decreases following $f(1)$.
Graph 2:
$f(x)$ has local maxima in $f(a)$, where $a \in [-1, 0]$ and $f(b)$, where $b \in [0, 3]$.
$f(x)$ has a local minimum in $f(0)$



Exercise 3

(a)

For this, we use the sum rule.

$$\frac{d}{dx}(x^3 + e^{2x}) = 3x^2 + 2e^{2x}$$

(b)

For this, we use the sum rule, and the chain rule.

$$\frac{d}{dx}(e^{x^2+3x^3}) = (2x + 9x^2)e^{x^2+3x^3}$$

(c)

For this, we use the quotient rule.

$$\frac{d}{dx}\frac{ln(x)}{x^2} = \frac{\frac{1}{x} * x^2 - ln(x) * 2x}{x^4} = \frac{x - ln(x) * 2x}{x^4}$$

(d)

For this, we use the chain rule, the sum rule and the rule of partial derivatives.

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

(e)

For this, we use the chain rule, the product rule, the sum rule and the rule of partial derivatives.

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

(f)

For this

(g)

Exercise 4

(a)

This task has been completed in the Jupyter Notebook, which has been handed in separately. Code and the images generated can be seen in the appendix.

(b)

This task has been completed in the Jupyter Notebook, which has been handed in separately. Code and the images generated can be seen in the appendix. When plotted side by side, it can be seen that the different partial derivatives are more sensitive to changes in their own direction, x and y respectively.

(c)

Increasing the step length, h seems to make edges stand out more, making them more readable to humans. However, this is only at small step lengths, at larger lengths the image seems to "split in two", meaning that it looks to the human eye as if there are two versions of the image, one much darker, the other much brighter.

(d)

This task has been completed in the Jupyter Notebook, which has been handed in separately. Code and the images generated can be seen in the appendix.

(e)

The gradient magnitude might be useful for making computers analyse different parts of images, and processing this data in an efficient way.

Appendix

Loading Packages

```
%matplotlib inline
# Allows viewing figures inline in the notebook
import numpy as np
# Numpy is a library for numerical computation
import matplotlib.pyplot as plt
# Matplotlib is a plotting library
from skimage import io
# skimage is an image processing library; its io module allows loading and saving images (
from matplotlib import cm
# import colormaps
import matplotlib
matplotlib.rcParams['figure.figsize'] = (20,8)
# Sets larger size for viewed figures
```

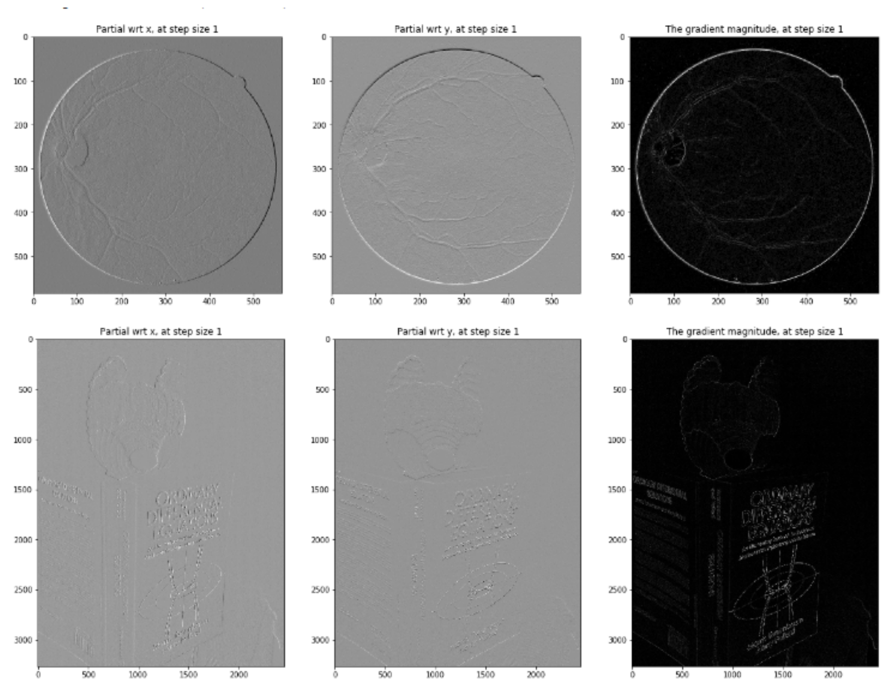
Exercise 4 a-b

```
IRet = io.imread('data/retinal.png').astype(float)
smartGirl = io.imread('data/smartgirl.jpg').astype(float)

lecPic = io.imread('lecture/flower.jpeg').astype(float)
Igray = np.mean(lecPic, axis=2)

#print("Original size:", IretGray.shape)
#print("Original size:", smartgirl.shape)

# Calculating the partial derivative with respect to x of the image
def xDiff(image, h, M):
    temp = (image[:,np.arange(2*h,M)] - image[:,np.arange(0,M-2*h)])

    # Padding with zeroes to make image the right shape
    xReduced = np.zeros(image.shape)
    xReduced[:temp.shape[0],:temp.shape[1]] = temp
    return xReduced

# Calculating the partial derivative with respect to y of the image
def yDiff(image, h, N):
    temp = (image[np.arange(0,N-2*h),:] - image[np.arange(2*h,N),:])

    # Padding with zeroes to make image the right shape
    yReduced = np.zeros(image.shape)
    yReduced[:temp.shape[0],:temp.shape[1]] = temp
    return yReduced

# Function for plotting and showing everything nicely
def partDiff(image, h):
    N, M = image.shape

    fig, ax = plt.subplots(1,2)

    ax[0].set_title('Partial wrt x, at step size %d' %h)
    ax[0].imshow(xDiff(image, h, M), cmap=cm.Greys_r)

    ax[1].set_title('Partial wrt y, at step size %d' %h)
    ax[1].imshow(yDiff(image, h, N), cmap=cm.Greys_r)

# Running everything with different step sizes
#partDiff(Igray,1)     # Testing with the picture from lectures
#partDiff(Igray,10)
#partDiff(Igray,25)
partDiff(IRet,1)
partDiff(IRet,50)
partDiff(smartGirl,1)
partDiff(smartGirl,4)
```

Exercise 4-d

```
def gradeMag(image, h):
    N, M = image.shape

    tempXDiff = xDiff(image, h, M)
    tempYDiff = yDiff(image, h, N)

    newImage = np.sqrt(tempXDiff**2 + tempYDiff**2)

    # Showing that the original image and result are the same size
    print("Original image size:\t", image.shape)
    print("New image size:\t\t", newImage.shape)

    fig, ax = plt.subplots(1,3)

    ax[0].set_title('Partial wrt x, at step size %d' %h)
    ax[0].imshow(tempXDiff, cmap=cm.Greys_r)

    ax[1].set_title('Partial wrt y, at step size %d' %h)
    ax[1].imshow(tempYDiff, cmap=cm.Greys_r)

    ax[2].set_title('The gradient magnitude, at step size %d' %h)
    ax[2].imshow(newImage, cmap=cm.Greys_r)
    #plt.title('The gradient magnitude')

gradeMag(IRet, 1)
gradeMag(smartGirl, 1)
#gradeMag(smartgirl, 100)
#gradeMag(smartgirl, 3)
```

```
#gradeMag(smartgirl, 4)
```


Partial wrt x, at step size 1
Partial wrt y, at step size 1
The gradient magnitude, at step size 1
Partial wrt x, at step size 1
Partial wrt y, at step size 1
The gradient magnitude, at step size 1