MASD Final Exam

MASD 2018
Department of Computer Science
University of Copenhagen

Eksamensnummer: 35

Version 1
Due: November 4th, 2018, 16:00

Contents

1. Exercise

(a)

i)

Here we use the chain rule $\left(\frac{d}{dx}f(x) = \frac{d}{dx}h(g(x)) * \frac{d}{dx}g(x)\right)$ and the rule for exponential functions: $\left(\frac{d}{dx}e^x = e^x\right)$

$$\frac{d}{dx}e^{2x^2+5} = 4xe^{2x^2+5}$$

ii)

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

First we use the product rule:$\left(\frac{d}{dx}f(x)g(x) = \frac{d}{dx}(f(x))g(x) + f(x)\frac{d}{dx}g(x)\right)$

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

We can then use the chain rule $\left(\frac{d}{dx}f(x) = \frac{d}{dx}h(g(x)) * \frac{d}{dx}g(x)\right)$ and the rule for exponential functions: $\left(\frac{d}{dx}e^x = e^x\right)$

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

We can now use the constant multiple rule: $\left(\frac{\partial}{\partial x}(cf(x) = c\frac{\partial}{\partial x}f(x)\right)$

$$\frac{\partial}{\partial y}(ln(x+y))e^{xy} + xln(x+y)e^{xy}\frac{\partial}{\partial y}y$$

We can now rewrite this to help us, and do the trivial derivative: $\left(\frac{\partial}{\partial x}(x^n) = nx^{n-1}\right)$

$$e^{xy}\left(\frac{\partial}{\partial y}ln(x+y) + xln(x+y)\right)$$

Now we have to use the chain rule again $\left(\frac{d}{dx}f(x) = \frac{d}{dx}h(g(x)) * \frac{d}{dx}g(x)\right)$ and the rule for the natural logarithm: $\left(\frac{\partial}{\partial x}ln(x) = \frac{1}{x}\right)$

$$e^{xy}\left(\frac{\frac{\partial}{\partial y}(x+y)}{(x+y)} + xln(x+y)\right)$$

We can now use the sum rule: $\left(\frac{\partial}{\partial x}(f(x)+g(x)) = \frac{\partial}{\partial x}f(x) + \frac{\partial}{\partial x}g(x)\right)$

$$e^{xy}\left(\frac{\frac{\partial}{\partial y}(x)+\frac{\partial}{\partial y}(y)}{(x+y)} + xln(x+y)\right)$$

We will now rewrite it so that it's easier to work with:

$$\frac{e^{xy}}{(x+y)}\left(\frac{\partial}{\partial y}(x) + \frac{\partial}{\partial y}(y) + (x+y)xln(x+y)\right)$$

We can now do the trivial derivatives $\left(\frac{\partial}{\partial x}c = 0\right)$ and $\left(\frac{\partial}{\partial x}(x^n) = nx^{n-1}\right)$:

$$\frac{e^{xy}}{(x+y)}\big(1 + (x+y)xln(x+y)\big)$$

We can now see that:

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

iii)

$$\frac{\partial}{\partial x}\frac{x^2y+y^2}{x^3-y}$$

First, we will rewrite the function to make it easier to work with:

$$\frac{\partial}{\partial x}\frac{y(x^2+y)}{x^3-y}$$

We can use the constant multiple rule: $\left(\frac{\partial}{\partial x}(cf(x) = c\frac{\partial}{\partial x}f(x)\right)$

$$y\frac{\partial}{\partial x}\frac{x^2+y}{x^3-y}$$

We then use the Quotient rule: $\left(\frac{\partial}{\partial x}\frac{f(x)}{g(x)} = \frac{f'(x)g(x)-f(x)g'(x)}{g(x)^2}\right)$

$$y\left(\frac{1}{(x^3-y)^2}(-(x^2+y)\frac{\partial}{\partial x}(x^3-y) + \frac{\partial}{\partial x}(x^2+y)(x^3-y))\right)$$

We can then rewrite this again:

$$\frac{-y((x^2+y)\frac{\partial}{\partial x}(x^3-y) - (x^3-y)\frac{\partial}{\partial x}(x^2+y))}{(x^3-y)^2}$$

We use the sum rule: $\left(\frac{\partial}{\partial x}(f(x)+g(x)) = \frac{\partial}{\partial x}f(x) + \frac{\partial}{\partial x}g(x)\right)$

$$\frac{-y\big((x^2+y)(\frac{\partial}{\partial x}(x^3) - \frac{\partial}{\partial x}y) - (x^3-y)(\frac{\partial}{\partial x}(x^2) + \frac{\partial}{\partial x}y)\big)}{(x^3-y)^2}$$

Now we can do the trivial derivatives of $\left(\frac{\partial}{\partial x}c = 0\right)$ and $\left(\frac{\partial}{\partial x}(x^n) = nx^{n-1}\right)$:

$$\frac{-y(3x^2(x^2+y) - 2x(x^3-y))}{(x^3-y)^2}$$

We will rewrite this to something more legible:

$$\frac{-xy(3x^3+3xy-2x^3+2y)}{(x^3-y)^2}$$

We can now say that:

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

iv)

$$\nabla(x^Tx) \text{ for } x = [x_1, ..., x_n]^T$$

First we will write this into something that's usable, using vector multiplication rules:

$$\nabla \sum_{i=1}^{n} x_i^T * x_i$$

This can be rewritten as the following, using the sum rule: $\left(\frac{\partial}{\partial x}(f(x)+g(x)) = \frac{\partial}{\partial x}f(x) + \frac{\partial}{\partial x}g(x)\right)$

$$\sum_{i=1}^{n} \frac{\partial}{\partial x_i}(x_i^T * x_i)$$

This cannot be reduced further, so we have the following:

$$\nabla(x^Tx) \text{ for } x = [x_1, ..., x_n]^T = \sum_{i=1}^{n} \frac{\partial}{\partial x_i}(x_i^T * x_i)$$

(b)

First we will explain the function $E(f) = \sum_{n=1}^{N}(y_n - f(x_n))^2$, then $f^* = argmin_{f\in\mathcal{F}}E(f)$

The function $E(f)$ goes through each coordinate of the dataset through the x-axis and compares the y-value of the given approximated function f to the y-value of the dataset. The difference of these is squared. The sum of this is taken for all coordinates. The result is then a measure of how close the function is to the dataset, where values closer to 0 mean it is more accurate

The way this is set up means that any difference between the given function and the dataset are exaggerated, meaning that even small deviations from the dataset result in large values of $E(f)$.

The second part $f^* = argmin_{f \in \mathcal{F}} E(f)$ chooses the function the matches the best, aka which function that returns the smallest value $E(f)$, in a given family of functions. The choice of $\mathcal{F}$ is a very important one in this function, since choosing one that does not fit the general curve of the dataset results in a function that is simply wrong. For example if we were to choose the family of linear functions, it is easy to see in the dataset given that past x-values of -3 and 3, it would be wildly inaccurate. At the same time, choosing an nth degree function would not work either, since there is too much noise.

(c)

To show that $argmin_a \mathbb{E}(f_a) = f^*$, we have to solve $\frac{\partial}{\partial a} \sum_{n=1}^{N} (y_n - x_n a)^2) = 0$ for a, when $f_a(x_n) = ax_n$. We start off by expanding the expression:

$$\frac{\partial}{\partial a} \sum_{n=1}^{N} (y_n^2 - 2x_n a y_n + (x_n a)^2) = 0$$

We can then use the sum rule to rewrite it to the following:

$$\sum_{n=1}^{N} \left(\frac{\partial}{\partial a} y_n^2 \right) - \frac{\partial}{\partial a} \Big(2x_n a y_n \Big) + \frac{\partial}{\partial a} \Big((x_n a)^2 \Big) = 0$$

Now we derive the functions:

$$\sum_{n=1}^{N} 0 - 2x_n y_n + 2a x_n^2 = 0$$

We can rewrite this into two separate sums:

$$\sum_{n=1}^{N} (2a x_n^2) - \sum_{n=1}^{N} (2x_n y_n) = 0$$

Rewrite it again, to try isolating a:

$$\sum_{n=1}^{N} 2a x_n^2 = \sum_{n=1}^{N} 2x_n y_n$$

We can put constant multiples outside of the sums:

$$2a \sum_{n=1}^{N} x_n^2 = 2 \sum_{n=1}^{N} x_n y_n$$

We reduce this:

$$a \sum_{n=1}^{N} x_n^2 = \sum_{n=1}^{N} x_n y_n$$

And isolate a:

$$a = \frac{\sum_{n=1}^{N} x_n y_n}{\sum_{n=1}^{N} x_n^2}$$

This is exactly what we wanted to show.

(d)

i)

To show that $E(f_W) = (Xw - y)^T(Xw - y)$, we will first write this out according to the error function given, part by part. First of all, we will define D as the highest value of d, to avoid confusion. Now, $f_w(x)$ can be written in a more useful way:

$$f_w(x_n) = w_1 x_n + w_2 x_n^2 + ... + w_D x_n^D = \sum_{d=1}^{D} w_d x_n^d$$

This can be set into the error function:

$$E(f_w) = \sum_{n=1}^{N} (y_n - f(x_n))^2 = \sum_{n=1}^{N} \Big(y_n - \Big(\sum_{d=1}^{D} w_d x_n^d \Big) \Big)^2$$

Now, we will show this backwards, by rewriting $(Xw - y)^T(Xw - y)$ to $E(f_w)$. First we will write the mathematical expression for each element in the resulting vector Xw:

$$Xw_n = \begin{bmatrix} x_1 & x_1^2 & \dots & x_1^d \\ x_2 & x_2^2 & \dots & x_2^d \\ \vdots & \vdots & \ddots & \vdots \\ x_N & x_N^2 & \dots & x_N^d \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} = w_1 x_n + w_2 x_n^2 + ... + w_D x_n^D = \sum_{d=1}^{D} w_d x_n^d$$

We can see that this is the same as $f_w(x_n)$, and will substitute that following. Since $\sum_{d=1}^{D} w_d x_n^d$ is only the expression for a single element in Xw, we can write the vector as thus:

$$Xw = \begin{pmatrix} f_w(x_1) \\ f_w(x_2) \\ \vdots \\ f_w(x_n) \end{pmatrix}$$

Then, $Xw - y$ must look like this:

$$(Xw - y) = \begin{pmatrix} f_w(x_1) - y_1 \\ f_w(x_2) - y_2 \\ \vdots \\ f_w(x_n) - y_n \end{pmatrix}$$

Now, to write $(Xw - y)^T(Xw - y)$, we can see the following:

$$\begin{pmatrix} f_w(x_1) - y_1 \\ f_w(x_2) - y_2 \\ \vdots \\ f_w(x_n) - y_n \end{pmatrix}^T \begin{pmatrix} f_w(x_n) - y_1 \\ f_w(x_n) - y_2 \\ \vdots \\ f_w(x_n) - y_n \end{pmatrix} = \sum_{n=1}^{N} \Big(f_w(x_n) - y_n \Big)^2$$

We know that $a^2 = (-a)^2$, so we can rewrite it as thus:

$$\sum_{n=1}^{N} \Big(- f_w(x_n) + y_n \Big)^2 = \sum_{n=1}^{N} \Big(y_n - f_w(x_n) \Big)^2$$

Which is exactly what we wanted to show:

$$E(f_w) = (Xw - y)^T (Xw - y) = \sum_{n=1}^{N} \Big(y_n - f_w(x_n) \Big)^2$$

ii)

For this part of the exercise, the proof has not been finished, but here is as far as I got:
We will use d and D the same way here as in the previous exercise, and will define $c = D$:

$$E(f_w) = \sum_{n=1}^{N} \left(y_n - \Big(\sum_{d=1}^{D} w_d x_n^d \Big) \right)^2$$

$$2X^T Xw - 2x^T y = 2 \begin{pmatrix} \sum_{n=1}^{N} \sum_{d=1}^{D} w_d x_n^d x_n^1 \\ \sum_{n=1}^{N} \sum_{d=1}^{D} w_d x_n^d x_n^2 \\ \vdots \\ \sum_{n=1}^{N} \sum_{d=1}^{D} w_d x_n^d x_n^c \end{pmatrix} - 2 \begin{pmatrix} \sum_{n=1}^{N} x_n^1 y_n \\ \sum_{n=1}^{N} x_n^2 y_n \\ \vdots \\ \sum_{n=1}^{N} x_n^c y_n \end{pmatrix}$$

$$2X^T Xw - 2x^T y = 2\Big(\sum_{n=1}^{N} \sum_{d=1}^{D} w_d x_n^d x_n^c \Big) - 2\Big(\sum_{n=1}^{N} x_n^c y_n \Big)$$

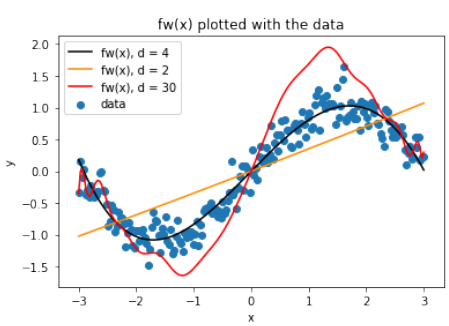
$$\nabla E(f_w) = \frac{\partial}{\partial w_d} \sum_{n=1}^{N} \left(y_n - \Big(\sum_{d=1}^{D} w_d x_n^d \Big) \right)^2 = 2\Big(\sum_{n=1}^{N} \sum_{d=1}^{D} w_d x_n^d x_n^c \Big) - 2\Big(\sum_{n=1}^{N} x_n^c y_n \Big)$$

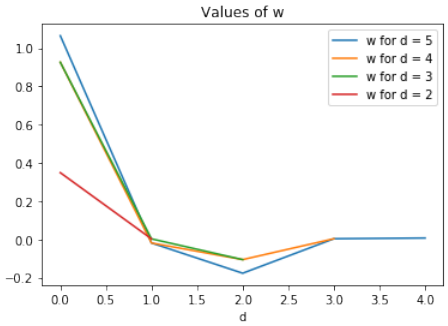
(e)

i)

ii)

The coding part of this Exercise has been completed in the attached Jupyter Notebook titled: "polyfit" When plotting for different values of d than 4, the function becomes more and more erratic and inaccurate the further away from 4 you get. At small differences, such as $d = 3$, you get only barely perceptible changes. For $d = 2$ however, you only get a straight line. For larger values, you get extremely erratic functions, that are significantly different, if you move just one x higher.





The most crucial parts of the code can be seen below. fw calculates the y-value of the function at the given points in the xvector. $calcw$ calculates each of the values of the vector w with a given dimension d:

```
def fw(w,x):
    retVal = 0
    for j in range(len(w)):
        retVal = retVal + w[j]*(x**(j+1))
    return retVal

def calcW(d, matrix):
    yVec = matrix[:,1].copy()
    xVec = data[:,0].copy()
    xmat = np.empty([len(matrix[:,0]), d])

    for i in range(len(xVec)):
        for j in range(d):
            xmat[i][j] = xVec[i]**(j+1)

    return inv(xmat.transpose() @ xmat) @ xmat.transpose() @ yVec
```

2. Exercise

(a)

i)

$$\int_1^4 x^3 - 4x^2 + \sqrt{x}dx$$

First we use the sum rule:

$$\int x^3 dx - \int 4x^2 dx + \int \sqrt{x}dx$$

These are trivial, so we calculate them separately and get the following:

$$\frac{x^4}{4} - \frac{4x^3}{3} + \frac{2}{3}x^{\frac{3}{2}}$$

Now we calculate these at the boundaries and subtract the result of the lower from the upper:

$$\left(\frac{4^4}{4} - \frac{4*4^3}{3} + \frac{2}{3}*4^{\frac{3}{2}}\right) - \left(\frac{1}{4} - \frac{4}{3} + \frac{2}{3}*1^{\frac{3}{2}}\right) = -16 - \left(\frac{-5}{12}\right)$$

Thus, we have found that the definite integral equals:

$$-16 - \left(\frac{-5}{12}\right)$$

ii)

$$\int ln(\sqrt{x})dx$$

First we rewrite this to the following:

$$\int \frac{ln(x)}{2}dx$$

We now take the constant out:

$$\frac{1}{2}\int ln(x)$$

We now do integration by parts:

$$\frac{1}{2}\left(xln(x) - \int 1dx\right)$$

The integral of a constant is $\int c = x$:

$$\frac{1}{2}\left(xln(x) - x\right)$$

Now we have to add a constant C, and we're done:

$$\frac{1}{2}\left(xln(x) - x\right) + C$$

iii)

$$\int x^2 e^{x^3} dx$$

Here, we have to do u-substitution:

$$\int x^2 e^u dx$$

We now have:

$$du = 3x^2 dx$$

$$du * \frac{1}{3x^2} = dx$$

With this we can substitute dx:

$$\int e^u x^2 \frac{1}{3x^2} du$$

This can be rewritten;

$$\int \frac{1}{3} e^u du$$

We take the constant out, and know that $\int e^x = e^x$, so we now have the result:

$$\frac{1}{3} \int e^{x^3} dx$$

We end up with:

$$\frac{1}{3} e^{x^3}$$

iv)

$$\int e^{-5r} dr$$

Here, we have to do substitution again:

$$\int e^u dx$$

We know that:

$$du = -5dx$$

We can then replace dx:

$$\int \frac{e^u}{-5} du$$

Now we have to take the constant out:

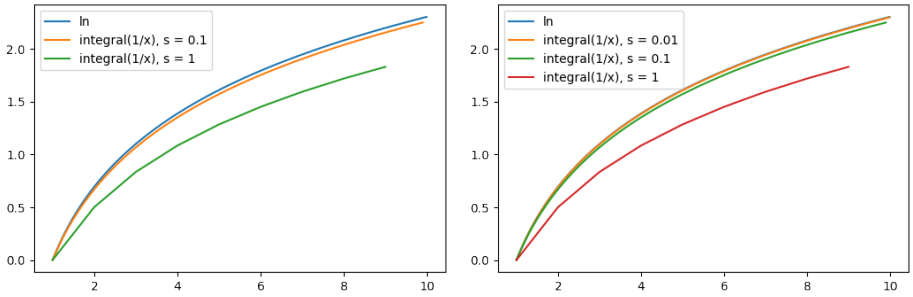
$$\frac{-1}{5} \int e^u du$$

And following the same rule as before $\int e^x = e^x$:

$$\frac{-1}{5} e^{-5r}$$

(b)

This Exercise has been completed in the attached Jupyter Notebook titled: "integration" Just in case, the resulting plots are given here. Note that there are two graphs to show that they follow $ln(x)$, because it is hidden behind the approximated function with step length 0.01



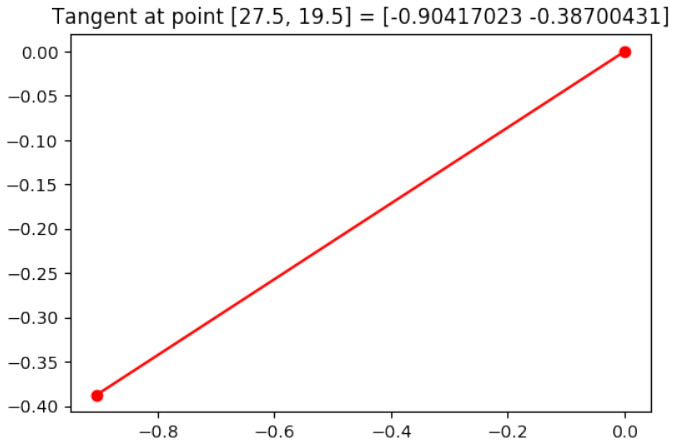
The most important bit of code is the following, which actually calculates the integral up to a given x:

```
def integrate(x, s):
    values = np.arange(1.0, x, s)
    retVal = [0]*len(values)
    for i in range(1, len(values)):
        if (i>0):
            retVal[i] = retVal[i-1] + s/values[i]
        else:
            retVal[i] = 0
    return retVal
```

(c)

i)

This Exercise has been completed in the attached Jupyter Notebook titled: "integration".
The resulting plot is given here:

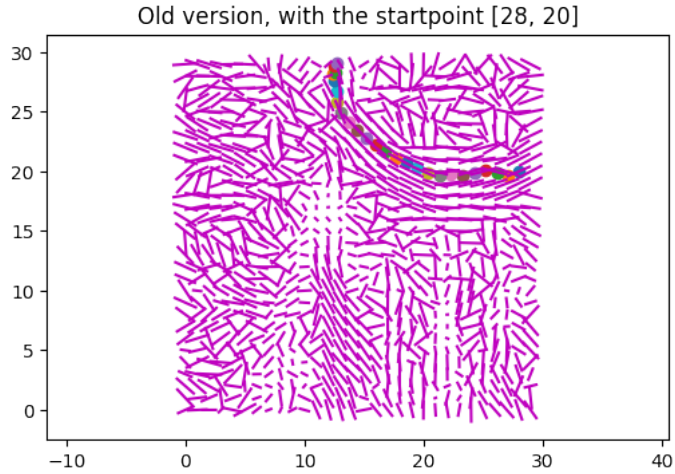


The most crucial part of the code is the following, which simply rounds the values given and returns the tangent in the given point:

```
def tanDir(floatPos, tanArr):
    intx = int(round(floatPos[0]))
    inty = int(round(floatPos[1]))
    return tanArr[intx,inty]
```

ii)

The coding part of this Exercise has been completed in the attached Jupyter Notebook titled: "integration".
When plotting the euler function on top of the tangential directions, you will see that the function follows the direction of the "flow". This means that in places where the vectors are all long and in a similar direction, the function will go fast in that direction, such as in the starting point $[28, 20]$. On the other hand, starting somewhere like $[12, 16]$, the vectors are all short and their direction seems at least sort of random, so each step will be short, and in an unpredictable direction. The resulting plot is given here:



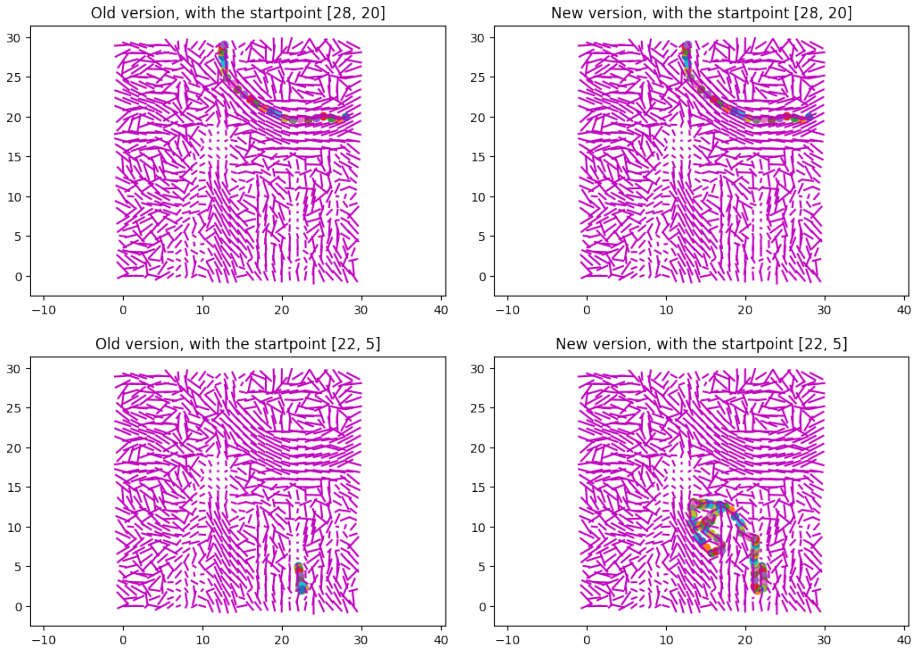
And the most crucial part of the code is the following, which creates a list of vectors, containing the coordinates for each step:

```
def euler(s, start, length, array):
    curPos = start.copy()
    retVal = [curPos]

    for i in range(length):
        # Checking if curPos is inside the given dataset
        if (curPos[0] < len(array)-1 and curPos[1] < len(array)-1 and curPos[0] > 0 and cu
            # Updating curPos with the tangents direction in the given point
            curPos += tanDir(curPos,array)
            retVal.append(curPos.copy())
        else:
            # Breaking out of the loop
            i = length
    return retVal
```

iii)

The coding part of this Exercise has been completed in the attached Jupyter Notebook titled: "integration".
To calculate which step would result in the smallest change in direction, you first have to figure out whether you came to this point via an inverted vector, if you did, you make sure that the vector you are reading for the current position is inverted as well. Then, you figure out whether the next vector should be inverted. This is done with some linear algebra, by calculating the angle between the current vector and the next vector in line. If the next vector should be inverted, we set a flag to make sure we invert it, as we checked for in the beginning. Then, we simply add the current vector the out current position.



The code for this bit has changed significantly, to take into account whether both the current, and the next vector is negative:

```
def newEuler(s, start, length, array):
    curPos = start.copy()
    retVal = [curPos]
    wasMinus = False
    for i in range(length):
        # Checking if curPos is inside the given dataset
        if (curPos[0] < len(array)-1 and curPos[1] < len(array)-1 and curPos[0] > 0 and cu

            # Determining which direction next step is
            if (wasMinus):
                curVec = -tanDir(curPos.copy(),array)
            else:
                curVec = tanDir(curPos.copy(),array)
            plusVector = tanDir((curPos.copy()+curVec),array)
            minusVector = -tanDir((curPos.copy()+curVec),array)
            # Updating curPos with the tangents direction in the given point
            if (vectorAngle(curVec,plusVector) < vectorAngle(curVec,minusVector)):
                wasMinus = False
            else:
                wasMinus = True

            curPos += curVec
            retVal.append(curPos.copy())
        else:
            # Breaking out of the loop
```

```
            i = length +1
    return retVal
```

iv)

One thing that stand out, seems to be the fact that plotting these tangents takes a surprisingly long time, even for a dataset as small as this one. With larger datasets, it would seem that this could be a problem.
Another problem with using the euler method is that while you can see where a fiber leads/comes from, from just a single element, you can only see it in one direction, that is, you cannot see where it would go further. Perhaps this could be fixed with an opposite function that instead of calculating for the positive value of the vector in the starting position, it would calculate for the negative value.

3. Exercise

(a)

The expression *pmf* stands for the "probability mass function".
The *pmf* of $X \sim Poiss(\lambda)$ can be written as follows, as per table 1.1 in the "MASD_essentials_ch1-updated.pdf" on Absalon:

$$p(x) = e^{-\lambda}\frac{\lambda^x}{x!}$$

This expression is used to calculate the probability of x events happening, when there is an average of λ events happening.

(b)

The expectation of Y, $\mathbb{E}Y$ can be written as follows, as per definition 1.8 in the "MASD_essentials_ch1-updated.pdf" on Absalon:

$$\mathbb{E}Y = \sum_y yp(y)$$

The variance of Y, $var(Y)$, can be written as follows, as per definition 1.10 in the "MASD_essentials_ch1-updated.pdf" on Absalon:

$$var(Y) = \mathbb{E}(Y - \mathbb{E}Y)^2 = \sum_y y^2 p(y) - (\sum_y yp(y))^2$$

To find $\mathbb{P}(a < V \leq b)$ we first have to make the observation that the cdf $F_V(x)$ is defined as being the probability that the variable V will be *less than or equal to x*, thus we can say that:

$$F_V(x) = \mathbb{P}(V \leq x)$$

This means that to calculate the probability of it being inside the interval $a < V \leq b$, we calculate $\mathbb{P}(V \leq b)$, and subtract $\mathbb{P}(V \leq a)$.

$$\mathbb{P}(a < V \leq b) = F_V(a) - F_V(b)$$

(c)

To prove that $\mathbb{E}Z = argmin_\alpha \mathbb{E}((Z-\alpha)^2)$, we will first write out what that means:

$$\mathbb{E}Z = \sum_z zp(z)$$

$$argmin_\alpha \mathbb{E}((Z-\alpha)^2) = \left(\frac{\partial}{\partial \alpha}\mathbb{E}((Z-\alpha)^2) = 0\right)$$

We will start by expanding the second expression:

$$\frac{\partial}{\partial \alpha}\mathbb{E}(Z^2 - 2Z\alpha + a^2) = 0$$

$$\frac{\partial}{\partial \alpha}\mathbb{E}(Z^2) - \mathbb{E}(2Z\alpha) + \mathbb{E}(a^2) = 0$$

Now we will write out the expressions in full:

$$\frac{\partial}{\partial \alpha}\Big(\sum_z z^2 p_z(z)\Big) - \frac{\partial}{\partial \alpha} 2\alpha \sum_z z p_z(z) + \frac{\partial}{\partial \alpha}\alpha^2 = 0$$

Now we will derive it with respect to α. $\frac{\partial}{\partial \alpha}\Big(\sum_z z^2 p_z(z)\Big) = 0$ because $\frac{\partial}{\partial x} c = 0$

$$0 - \frac{\partial}{\partial \alpha} 2\alpha \sum_z z p_z(z) + \frac{\partial}{\partial \alpha}\alpha^2 = 0$$

Now we derive the second part of the expression, using the rule $\frac{\partial}{\partial \alpha} c\alpha = c$

$$\frac{\partial}{\partial \alpha} 2\alpha \sum_z z p_z(z) = 2\sum_z z p_z(z)$$

To derive the the third part, we use the rule that $\frac{\partial}{\partial \alpha}\alpha^n = n\alpha^{n-1}$

$$\frac{\partial}{\partial \alpha}\alpha^2 = 2\alpha$$

Now we can add all this into the expression, and end with this:

$$-2\sum_z z p_z(z) + 2\alpha = 0$$

We can rewrite this to the following:

$$\sum_z z p_z(z) = \alpha$$

This is what we wanted to show, now we have:

$$argmin_\alpha \mathbb{E}((Z-\alpha)^2) = \sum_z z p_z(z) = \mathbb{E}Z$$

4. Exercise

(a)

For this, I assume that the given values for x and y are all the values possible. If that assumption is true, then the final joint pmf will look as follows:

$p(x,y)$	y=-1	y=1	y=3
x=-1	0.1	0.1	0.4
x = 1	0.3	0.05	0.05

The reason for this is because the joint pmf must be equal to 1. So we simply calculate:

$$1 - (0.1 + 0.1 + 0.3 + 2(0.05)) = 0.4$$

(b)

To compute $\mathbb{E}(X+Y)$ we have to make the following observation:

$$\mathbb{E}(X+Y) = \mathbb{E}X + \mathbb{E}Y$$

This means that we simply have to compute:

$$\mathbb{E}X + \mathbb{E}Y = \sum_x x p_X(x) + \sum_y y p_Y(y)$$

To find the marginal pmfs $p_X(x)$ and $p_Y(y)$ we use the following calculations:

$$p_X(x) = \sum_y p_{X,Y}(x,y)$$

$$p_Y(y) = \sum_x p_{X,Y}(x,y)$$

First we will find $\mathbb{E}X$, then $\mathbb{E}Y$:

$$\mathbb{E}X = -1(0.1 + 0.1 + 0.4) + 1(0.3 + 0.05 + 0.05) = -0.2$$

$$\mathbb{E}Y = -1(0.1 + 0.3) + 1(0.1 + 0.05) + 3(0.4 + 0.05) = 1.1$$

Then we simply write:

$$\mathbb{E}(X+Y) = -0.2 + 1.1 = 0.9$$

This should be interpreted as expecting to make back 90% of the money put in.

(c)

First we will compute the pmf p_Z of $Z = X * Y$.
To do this, first we will define what p_Z looks like:

$$p_Z(z) = \mathbb{P}(Z = z)$$

We can see from the table from Exercise 4a that there are six possible values for Z, since there are 2 values for X and 3 for Y. These values are the following:

$X * Y$	$y = -1$	$y = 1$	$y = 3$
$x = -1$	1	-1	-3
$x = 1$	-1	1	3

Now we simply compute $\mathbb{P}(X * Y)$ for each value of x and y, and get the following table:

$p_Z(z)$	$\mathbb{P}(X * Y = z)$
$z = -3$	0.4
$z = -1$	0.4
$z = 1$	0.15
$z = 3$	0.05

To figure out if X and Y are independent, we have to look at their covariance, which is defined as thus:

$$cov(X, Y) = \mathbb{E}XY - \mathbb{E}X\mathbb{E}Y$$

We start off by finding $\mathbb{E}XY$:

$$\mathbb{E}XY = \sum_{x,y} xyp(x, y)$$

I have done this calculation in the attached Jupyter Notebook named: "Calculations" since it would be tedious to do by hand. It has given me the following:

$$\mathbb{E}XY = -1.3$$

We have already calculated $\mathbb{E}X$ and $\mathbb{E}Y$ further up, in exercise 4b. So we can write the following:

$$cov(X, Y) = (-1.3) - (-0.2) * 1.1 = -1.08$$

Since we now know that the covariance $cov(X, Y) \neq 0$, we also know that X and Y are dependant on each other.

(d)

To show that $cov(V, W) = 0 \implies V$ and W are independent for Bernoulli distributions of V and W, we first have to look at the definition of independence, as per Definition 1.15 in "MASD_essentials_ch1-updated.pdf" on Absalon. I will rewrite it slightly so it fits the exercise at hand:

$$\mathbb{P}(X \in \{0,1\})\mathbb{P}(Y \in \{0,1\}) = \\ \mathbb{P}(X \in \{0,1\}, Y \in \{0,1\}) \text{ for all } x, y \implies \text{mutual independence}$$

So now we can see that we need to show that:

$$\mathbb{E}VW - \mathbb{E}V\mathbb{E}W = 0 \implies \\ \mathbb{P}(V \in \{0,1\}, W \in \{0,1\}) - \mathbb{P}(V \in \{0,1\})\mathbb{P}(W \in \{0,1\}) = 0 \implies \\ \mathbb{P}(V \in \{0,1\})\mathbb{P}(W \in \{0,1\}) = \mathbb{P}(V \in \{0,1\}, W \in \{0,1\})$$

We know that for Bernoulli distributions $\mathbb{E}V = \theta$ since:

$$\mathbb{E}V = \sum_v vp(v) = \mathbb{P}(V = 0) * 0 + \mathbb{P}(V = 1) * 1 = \theta = \mathbb{P}(V \in \{0,1\})$$

The same holds for $\mathbb{E}W = \phi$:

$$\mathbb{E}W = \sum_w wp(w) = \mathbb{P}(W = 0) * 0 + \mathbb{P}(W = 1) * 1 = \theta = \mathbb{P}(W \in \{0,1\})$$

To find $\mathbb{E}VW$, we have to write up the joint pmf. For brevity we will define $\mathbb{P}(V = 1) = a$, $\mathbb{P}(W = 1) = b$ and $\mathbb{P}(V = 1, W = 1) = c$. We can now say the following:

$$\mathbb{P}(V = 1, W = 0) = a - c \in [0,1]$$
$$\mathbb{P}(V = 0, W = 1) = b - c \in [0,1]$$
$$\mathbb{P}(V = 0, W = 0) = 1 - a - b - c \in [0,1]$$

We can now rewrite the pmf slightly to show that $\mathbb{P}(VW \in \{0,1\}) = \mathbb{P}(V \in \{0,1\})\mathbb{P}(W \in \{0,1\})$

$$p(v,w) = \begin{cases} \mathbb{P}(V = 0, W = 0) & = 1 - a - b - ab & = (1-a)(1-b) & = \mathbb{P}(V = 0) * \mathbb{P}(W = 0) \\ \mathbb{P}(V = 0, W = 1) & = b - ab & = (1-a)b & = \mathbb{P}(V = 0) * \mathbb{P}(W = 1) \\ \mathbb{P}(V = 1, W = 0) & = a - ba & = a(1-b) & = \mathbb{P}(V = 1) * \mathbb{P}(W = 0) \\ \mathbb{P}(V = 1, W = 1) & = ab & = ab & = \mathbb{P}(V = 1) * \mathbb{P}(W = 1) \end{cases}$$

We can now see that for all values of V and W, the following must be true:

$$\mathbb{E}VW - \mathbb{E}V\mathbb{E}W = 0 \implies p(v,w) - p(v)p(w) = 0 \implies p(v,w) = p(v)p(w)$$

And since $cos(V,W) = \mathbb{E}VW - \mathbb{E}V\mathbb{E}W$, the proof is finished.