Homework 1

Joakim Ginste  
19970614-7155

**Question 1:**

1. Results from the MATLAB codes, whilst using single precision are given below, if not single precision would have been used here, we would have gotten D = E due to that a round of error not happening as MATLAB use up to 16 value figures (double precision):

D = 1.333334088325501  
E= 1   
Error = 0.749999582767487  
Error in percent = 25.0000417232513%

1. For this assignment single precision was not set so Matlab’s standard was used, “double precision”.

D = 1.333334100000000  
E = 1.375000000000000  
Error = 0.969697527272727  
Error in percent = -3.1249407031591%

1. Comparing these error results against the machine epsilon for single precision (2^-23) and double precision (2^-52) we see that they are a lot bigger than the round of error that machine epsilon gives the maximum value for. But this round of error is the reason for these errors!

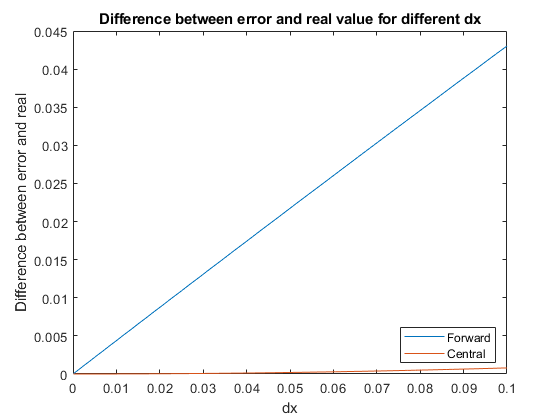
**Question 2:**

Machine epsilon when using Matlab is given by eps(1) = 2.220\*10^-16 = 2^(-52)

The value for the first derivative of cos(x) using forward and central differencing at x = 0.5 radians depends on the step size. An vector for both these values are given in the MATLAB code under (f\_x\_central) and (f\_x\_forward)

1.

**Plot for the total error using both forward and central differencing.**

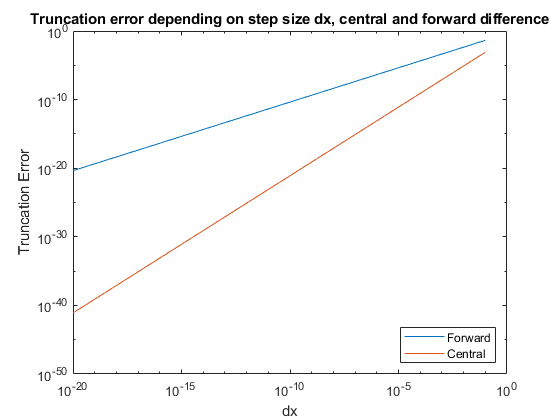


2. From Taylor series of degree 2 we get the truncation error for **forward differentiation** (HOT from degree 3 or higher are not included in this error.):

With the same logic we get for **central differentiation** (HOT from degree 4 or higher are not included in this error)**:**

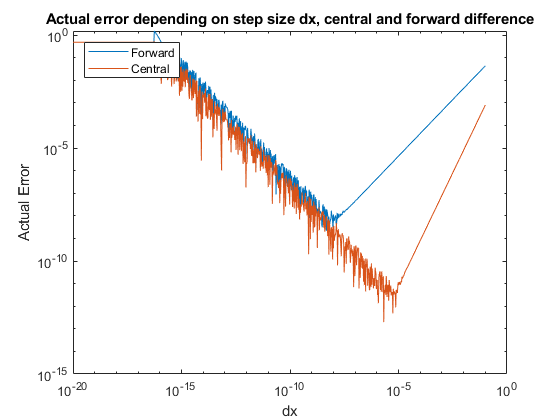
Both values have been given as **absolute** values in the plots, as the importance is to compare their magnitude to each other. As can be seen in the figure below central difference have a much lower error for all step sizes. This is due to its dependence of dx is of a degree higher than for forward differencing.

**Plot of truncation error for both central and forward differencing.**



1. **Actual error for Central and Forward differencing. (Magnitude)**

In this plot we see a minimum value for the actual error at 10^-10 (central differencing) and 10^-7 (forward differencing). The minimum value is at this point due to the combined error from truncation error and the round off error. As the round of error is less at larger number for dx, whilst the truncation error is less at smaller values for dx. The minimum value of the total error exists somewhere between the smallest and largest value for dx for this reason.



**Question 3:**

**Disclaimer:** I wrote my MATLAB code by changing the inner matrix with the unknow temperatures to a vector instead and therefore my codes I quite messy, there were a lot of edge cases (all edge points needed their own for loops, as well as all the outer rows and columns that were close to the boundary condition). I still think my method worked well for any inputs of dx that lead to number of rows/columns that are working for 3 or 5 point central differencing.

The MATLAB code for this question is also divided into the A and B case where we had dx = dy, and dy = 0.5dx respectively due to that I got new functions for the central differencing formula that required to rework the coefficients in the A matrix a bit.

For 3 and 5 point central differencing, the equation that were used are given below:

With dx = dy these two equations will be simplified by adding the equations together under the same denominator and multiplying up the denominator the left-hand side (that is equal to 0) to:

The coefficient before the different temperature points is put into the A matrix, used to solve for the sought of temperature at point Tx,y.

For dy = 0.5dx we do the same thing, but we must change the dy = 0.5dx and that will affect the coefficients before each point as can be seen in the equations below:

**A and B results:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method** | **3 Points** | | **3 Points** | |
| **Points** | dx = dy = 0.25 | dx = dy = 0.125 | dx = 0.25  dy = 0.5\*dy | dx = 0.0125  dy =0.5\*dx |
| 1 [°C] | **19.8942** | **19.8569** | **19.8969** | **19.8575** |
| 2 [°C] | **20.6317** | **20.525** | **20.6052** | **20.5179** |
| 3 [°C] | **37.3942** | **37.3569** | **37.4717** | **37.3762** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method** | **5 Points** | | **5 Points** | |
| **Points** | dx = dy = 0.25 | dx = dy = 0.125 | dx = 0.25  dy = 0.5\*dy | dx = 0.0125  dy =0.5\*dx |
| 1 | **20.3646** | **19.9706** | **20.1028** | **19.9082** |
| 2 | **20.6374** | **20.5204** | **20.5675** | **20.5053** |
| 3 | **37.3735** | **37.3507** | **37.3897** | **37.3517** |

Comparing these results, the most accurate one should be the one with the most points that the steel plate is divided into. That is achieved with 5 points central differencing with dx = 0.0125 and dy = dx\*0.5. The contour plots for that solution are given below. Contour plots were made for all points in the MATLAB codes also.

**Contour plot for 5 points central differencing with dx = 0.0125 and dy =0.5\*dx**

