Vaasan yliopisto
UNIVERSITY OF VAASA

REPORT

FINAL HOMEWORK

PEDRO MIRANDA – x102690

TLTE.3120

COMPUTER SIMULATION IN COMMUNICATION AND SYSTEMS

EXCHANGE STUDENT
2014/2015

31-01-2014

Index

1. Introduction

The present document reports the final homework of the course TLTE.3120 – Computer simulation in communication and systems.

During this course were teach two fundamental tools of the computer software Matlab:

- Work with code scripts in the Matlab environment.
- Work with Simulink plugin.

This final homework can be split in two fundamental tasks.

The first task reflects the knowledge of the student about code scripting. The second is a solution to one problem using Simulink.

In a summary, the task of code scripting is related with images processing and to the solution of the classical travelling salesman problem. The second task is related with the construction in Simulink of two models of ADC using of delta sigma modulation of second and third order.

2. Matlab programming task

2.1. Task definition

The objective of the first part of the final homework is to create a program to solve a problem based on several tasks described on a program template.

The problem can be divided in 4 essential parts:

- Matlab process to read images.
- Calculation of matrix that reflect differences between images.
- Solve the classical travelling salesman problem (TSP) using the matrix calculated in the previous step.
- Display the results of the TSP.

Furthermore, the solution presented was developed in a way that it is possible to compare images with different sizes (function differ_weighted__mean, developed in TASK6) and also coloured images (TASK7).

2.2. Adopted approach and solutions

2.2.1. Process to read images

The process of read images is part of task one of the programming template.

In this task is possible to read:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TASK 1:
%           Change the previous image reading part so that you read
%           all the 20 object images in a loop.
%           Hint: you will need string parsing to parse name
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

To solve this task were used the functions strcat – function that permits the concatenation of strings – and the function eval – that evaluates the MATLAB® code in the string expression.

In order to solve TASK7 – read colour images – was added to the code the function imfinfo and info.ColorType, they were used to evaluate if the image is truecolor or greyscale.

The solution achieved is the following one:

```
n_images = 20; %number of images to read

for k = 1:n_images
    %read the image
    filename = strcat(['obj' num2str(k) '.png']);
    eval(['IM' num2str(k) '=imread(filename);' ]);

    %convert rgb images in grayscale (TASK 7)
    info = imfinfo(['obj' num2str(k) '.png']);
    if(info.ColorType == 'truecolor')
        eval(['IM_Color_' num2str(k) '=(IM' num2str(k) ');']);
        eval(['IM' num2str(k) '=rgb2gray(IM' num2str(k) ');']);
    end
end
clearvars filename k info; %clear temporary variables
```

With this code is possible to read the 20 objects, without any problem. Changing the number of n_images is possible to read a different number of images since images are saved with the same philosophy of filename.

2.2.2. Calculation of matrix that reflect differences between images.

This part of the program was divided in two tasks:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  TASK 2:   Change the previous image comparison part so that you
%            compare all 20 object images against each other inside
%            loops
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  TASK 3:   Now you have the differences between images, next fill
%            the bottom of the table too
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

To solve task 2 was used the following code:

```
% Calculate the difference between images as grayvalue difference for
%diff(IMx, IMy) where y is a numerical value bigger than x.
for i=1:n_images
    for j=1:n_images
        if i<j
            diff(i,j)=differ(eval(['IM'    num2str(i)]),    eval(['IM'
num2str(j)]));
        end
    end
end
clearvars i j; %clear temporary variabels
```

This double loop compares all images with each other with the function differ.

To avoid calculations of diff(y,x) when diff(x,y) is already calculated was added to the code a if clause, that just permits the calculation of differ when i<j.

The only difference of this code on task 6 is the differ was changed to differ_weighted_mean.

To solve task 3 was used the following code:

```
%Sum of table with its transpose to obtain the final diff table
diff = diff + diff.';
```

This code allows to fill the table with all the values necessary for solve the travelling salesman problem.

The code assign to diff matrix the result of diff matrix and its transpose.

2.2.3. Solve the classical travelling salesman problem (TSP) using the matrix calculated in the previous step.

To solve the TSP were used two external functions from the website mathworks:

- tspo_ga
- tsp_ga

Both functions are genetic algorithms. The first one is a solution for the TSP open case – start from one image, but end to some other – and the second one is the solution for the TSP close case –start and end at the same image.

The code exposed in the following lines reflects the approach and solution adopted:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% a) open loop (start from one image, but end to some other)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%This solution uses the function TSPO_GA that can be found in the
following
%webpage:
%http://www.mathworks.com/matlabcentral/fileexchange/21196-open-
traveling-salesman-problem-genetic-algorithm/content/tspo_ga.m

%Resulution
% Change the defaults values of the configuration formula
userConfig                          =                          struct('xy',
diff,'popSize',100,'numIter',1e5,'showProg',false,'showResult',false,'
showWaitbar',true);
%Application of the formula
resultStruct_open = tspo_ga(userConfig);

clearvars userConfig %clear temp variabels

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%b) closed loop (start and end at the same image)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%This solution uses the function tsp_ga that can be found in the
%following webpage:
%http://www.mathworks.com/matlabcentral/fileexchange/13680-traveling-
salesman-problem-genetic-algorithm

%Resolution

% Change the defaults values of the configuration formula
userConfig                          =                          struct('xy',
diff,'popSize',100,'numIter',1e5,'showProg',false,'showResult',false,'
showWaitbar',true);
%Application of the formula
resultStruct_close = tsp_ga(userConfig);

clearvars userConfig %clear temp variables
```

All the results of both functions were saved in two structures:

- resultStruct_open
- resultStruct_close

This results were used to display the optimal routes in two figures.

2.2.4. Display the results of the TSP

To display the results were used two figures, in each figure were display the images following the optimal routes computed before. The mechanisms “figure” and “subploting” were used to display the images correctly.

The following code was the approach used to reach this objective:

```
%Display TSP_open
figure

%Dimensions of the plot
x_table = 9;
y_table = 4;

%NOTE:This table was optimized for 20 images, you should optimize this
%table case you want to use this program with a different number of
images

for k = 1:n_images
    %Chosing the subplot
    if k <  n_images/2
        subplot(y_table,x_table,k);
    elseif k == n_images/2
        subplot(y_table,x_table,18);
    elseif k == n_images/2 + 1
        subplot(y_table,x_table,27);
    elseif k > n_images/2 + 1
        subplot(y_table,x_table,(y_table  *  x_table)-(k-(n_images/2  +
2)));
    end
    %Get information about images
    info   =   imfinfo(['obj'   num2str(resultStruct_open.optRoute(k))
'.png']);
    %Choose the filename of the object to be read
    if info.ColorType == 'truecolor'
        filename                 =                 strcat(['IM_Color_'
num2str(resultStruct_open.optRoute(k))]);
    else
        filename                     =                     strcat(['IM'
num2str(resultStruct_open.optRoute(k))]);
    end
    %Show image
    imshow(eval(filename));
    %Puts titles in the subplots
    mt(k)                    =                    title(strcat(['Obj'
num2str(resultStruct_open.optRoute(k))]));
    hold on
```

```
end
%Set name of the figure
set(gcf,'Name','TSP_Open');

%Clean temporary variables
clearvars k filename mt info x_table y_table;


%Display TSP_open
figure

%Dimensions of the plot
x_table = 9;
y_table = 3;
first_break = x_table * (y_table - 1);

%NOTE:This table was optimized for 20 images, you should optimize this
%table case you want to use this program with a different number of
images

for k = 1:n_images
    %Choose subplot
    if k<10
        subplot(y_table,x_table,k);
    elseif k==(n_images/2)
        subplot(y_table,x_table,first_break);
    elseif (k > (n_images/2) && k <n_images)
        subplot(y_table,x_table,first_break+(n_images-k));
    elseif k == n_images
        subplot(y_table,x_table,(n_images/2));
    end
    %Get information about images
    info   =   imfinfo(['obj'   num2str(resultStruct_close.optRoute(k))
'.png']);
    %Choose the filename of the object to be read
    if info.ColorType == 'truecolor'
        filename                 =                 strcat(['IM_Color_'
num2str(resultStruct_close.optRoute(k))]);
    else
        filename                       =                       strcat(['IM'
num2str(resultStruct_close.optRoute(k))]);
    end
    %Show image
    imshow(eval(filename));
    %Puts titles in the subplots
    mt(k)                      =                    title(strcat(['Obj'
num2str(resultStruct_close.optRoute(k))]));
    hold on
end
%Set name of the figure
set(gcf,'Name','TSP_Close');

%Clean temporary variables
clearvars k filename mt info x_table y_table first_break n_images;
```

2.2.5. Task 6 – Implement another function to compare images

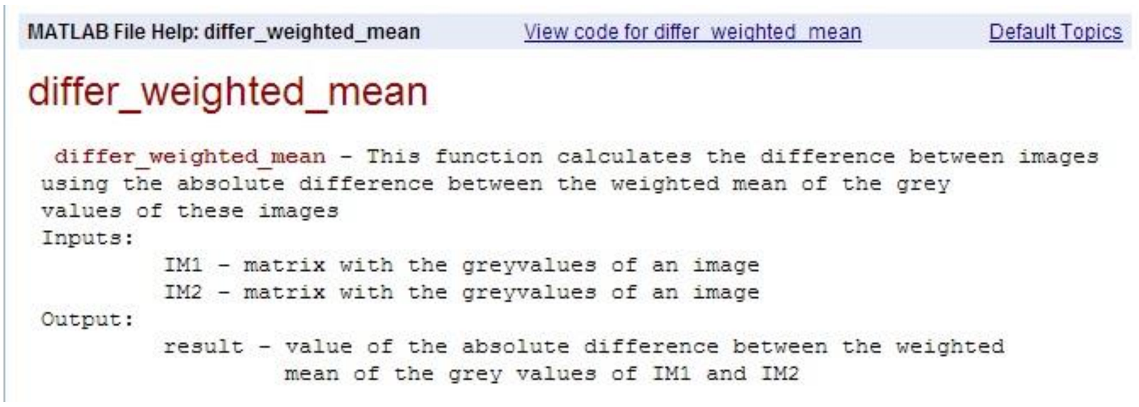
The function differ_weighted_mean was implemented to compare images.

This function compares images using the absolute difference between the weighted mean of the grey values of these images.

The code of this function is the following one:

```matlab
function [result] = differ_weighted_mean(IM1, IM2)
% DIFFER_WEIGHTED_MEAN - This function calculates the difference between
images
%using the absolute difference between the weighted mean of the grey
%values of these images
%Inputs:
%          IM1 - matrix with the greyvalues of an image
%          IM2 - matrix with the greyvalues of an image
%Output:
%          result - value of the absolute difference between the
%                   weighted mean of the grey values of IM1 and IM2

%Determine the width and height of the matrixes that you want to compare
[IM1_height, IM1_width] = size(IM1);
[IM2_height, IM2_width] = size(IM2);

%Sum of all the values of the matrix that reflects the images
IM1_sum = sum(IM1(:));
IM2_sum = sum(IM2(:));

%Mean of the values normalized with the size of the image
IM1_grey_mean = IM1_sum / (IM1_height * IM1_width);
IM2_grey_mean = IM2_sum / (IM2_height * IM2_width);

%Comparison of the absolute value of the means
result = abs(IM1_grey_mean - IM2_grey_mean);
```

In case of the user ask for information about this function the output will be the following one:



MATLAB File Help: differ_weighted_mean View code for differ_weighted_mean Default Topics

differ_weighted_mean

```
 differ_weighted_mean - This function calculates the difference between images
using the absolute difference between the weighted mean of the grey
values of these images
Inputs:
         IM1 - matrix with the greyvalues of an image
         IM2 - matrix with the greyvalues of an image
Output:
         result - value of the absolute difference between the weighted
                  mean of the grey values of IM1 and IM2
```

2.3. Final results and outputs

The final results and outputs were the following ones:

TSP open grey images solution vector: 3 6 5 9 13 12 10 11 15 17 14 4 1 2 7 8 16 20 18 19

Distance registered on solution vector: 20016976.2371761

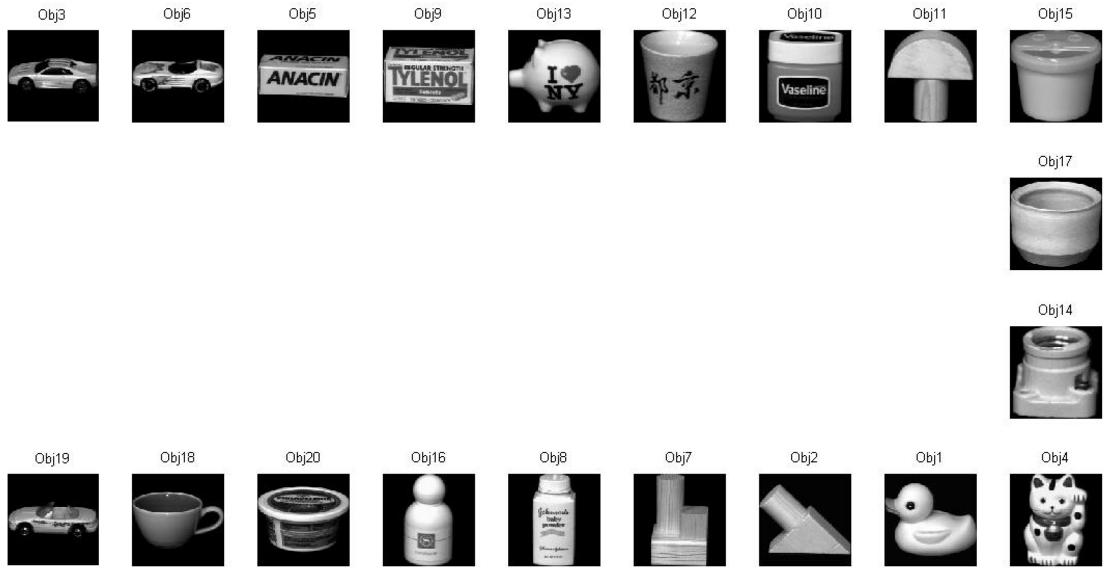


fig. 1 - TSP open grey images solution

____

TSP close grey images solution vector: 7 8 16 19 18 20 6 3 5 9 13 12 10 11 15 17 14 4 1 2

Distance registered on solution vector: 23377215.5894578

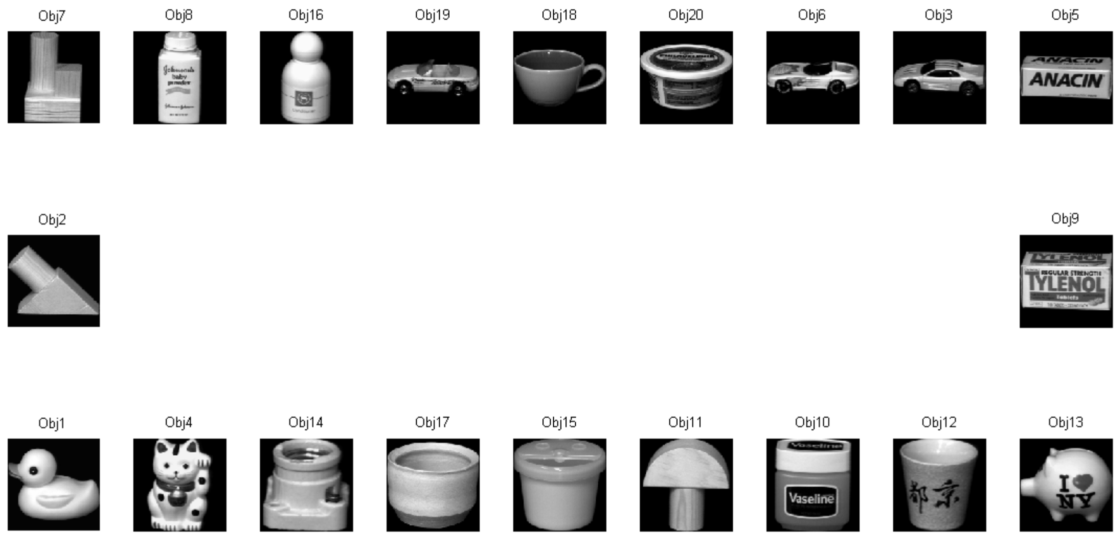


fig. 2 - TSP close grey solution

TSP open colour images with differ function, solution vector: 2 1 4 3 5 6 7 8 14 17 18 15 16 13 10 11 12 9 19 20

Distance registered on solution vector: 34619188.7699312

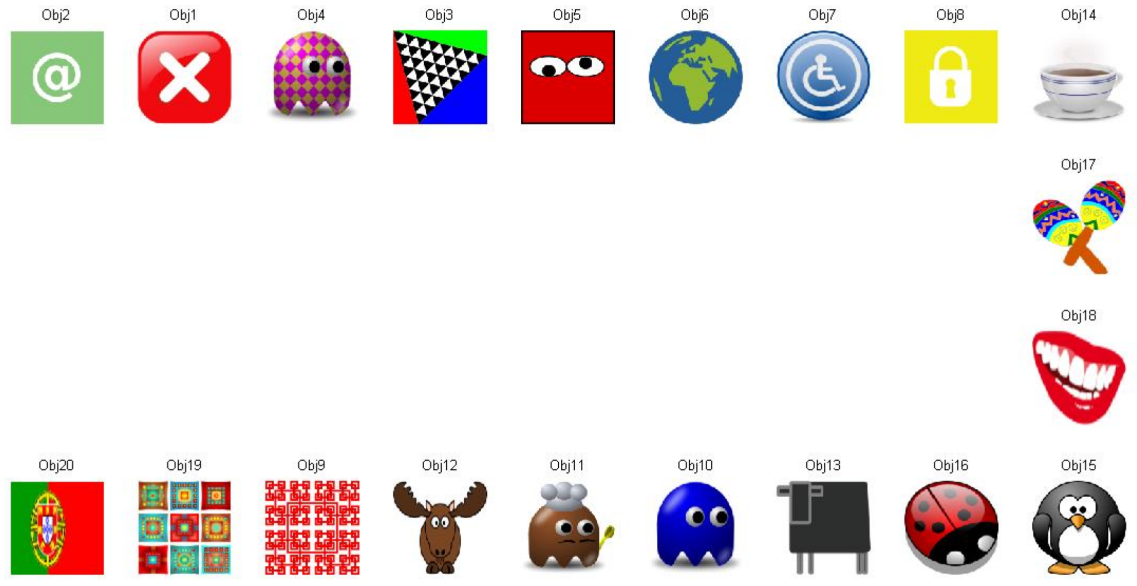


fig. 3 - TSP open colour, with differ function, solution

____

TSP close colour images differ function, solution vector: 3 5 6 9 12 11 10 13 16 20 19 15 18 17 14 7 8 2 1 4

Distance registered on solution vector: 37977649.3295302

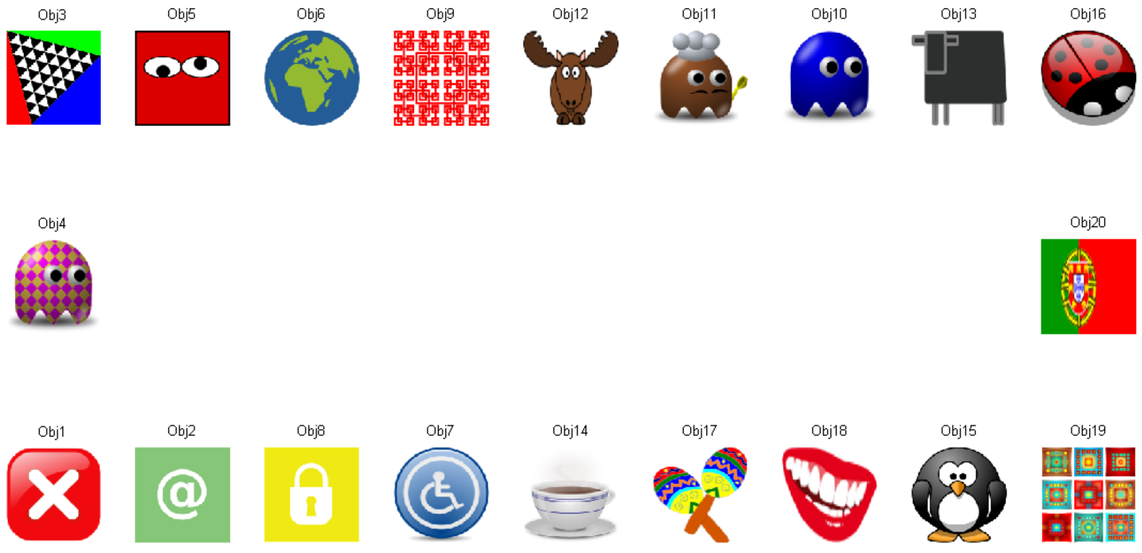


fig. 4 - TSP close colour, with differ function, solution

TSP open colour images with differ_weighted_mean function, solution vector: 8 2 7 12 4 15 9 6 19 17 14 10 11 20 3 1 16 13 5 18

Distance registered on solution vector: 837.971933718382

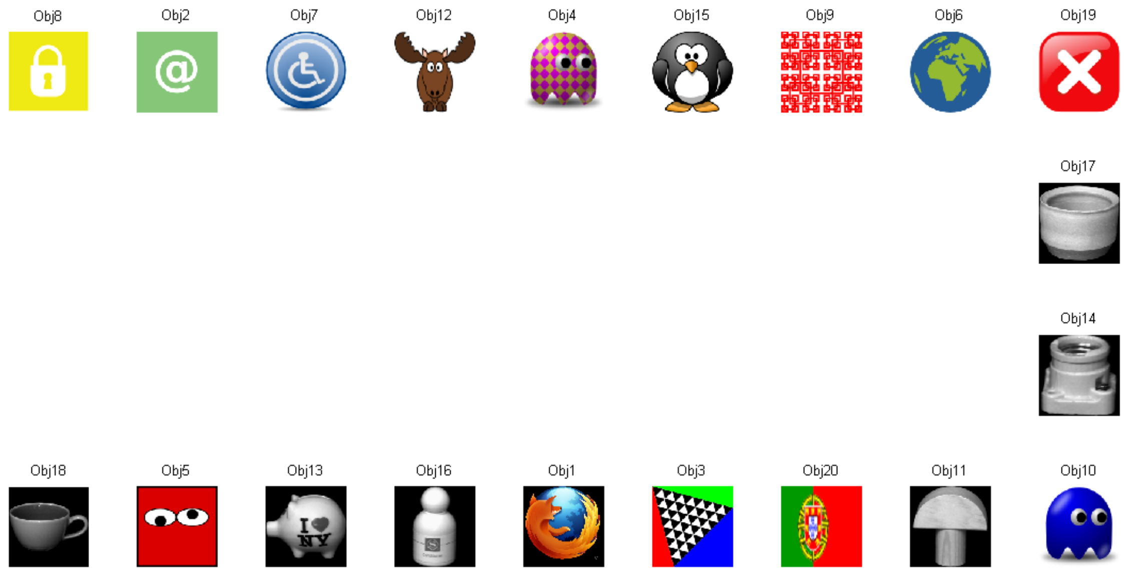


fig. 5 - TSP open colour, with differ_weigthed_mean function, solution

____

TSP close colour images differ_weighted_mean function, solution vector: 19 6 9 15 4 12 7 2 8 18 5 13 16 1 3 20 11 10 14 17

Distance registered on solution vector: 1241.24747694215

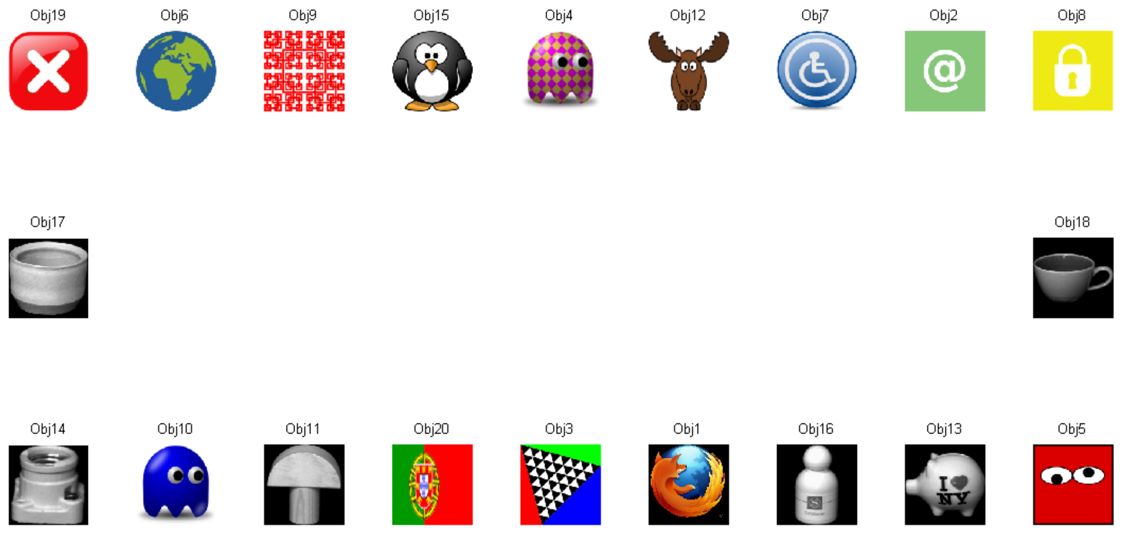


fig. 6 - TSP close colour, with differ_weighted_mean function, solution

Notice that this solution has objects with grey values and true colour values.

Furthermore, there are images with different sizes (ex: obj1 = 1233x1233 pixels and obj2 = 128x128 pixels).

3. Simulink Modelling Task

3.1. Task definition

The objective of the Simulink modelling task is to create a second and third degree delta-sigma analog-to-digital converter (ADC) models with Simulink, using as base of work a first degree delta-sigma ADC.

3.1.1. Definition of delta-sigma modulation

Delta-sigma modulation is digital signal processing. This process is used for:

- encoding analog signals into digital signals
- transfer higher resolution digital signals into lower resolution digital signals as part of the process to convert digital signals into analog.

In delta sigma modulation the change in the signal is encoded. In this way the result is a stream of pulses instead of a stream of numbers like in PCM conversion. Due to that, it is possible to decrease the quantization error noise.

Furthermore, the accuracy of the modulation is improved by passing the digital output through a 1-bit DAC and adding (sigma) the resulting analog signal to the input signal, thereby reducing the error introduced by the delta-modulation.

3.2. Adopted approach and solutions

Using as reference the ADC model in fig. 7 were built the models of the ADC in fig. 8 and fig. 9.

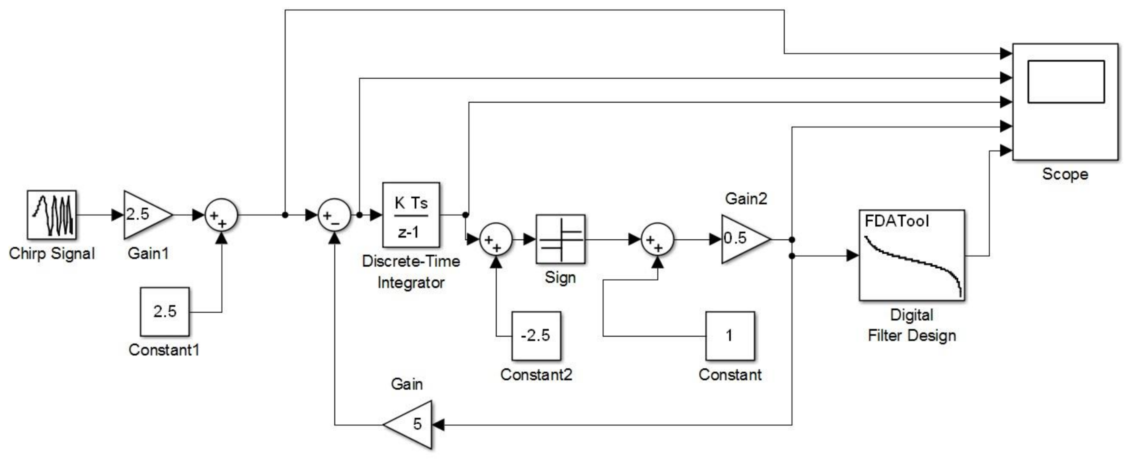


fig. 7 – ADC with first order delta-sigma modulation model

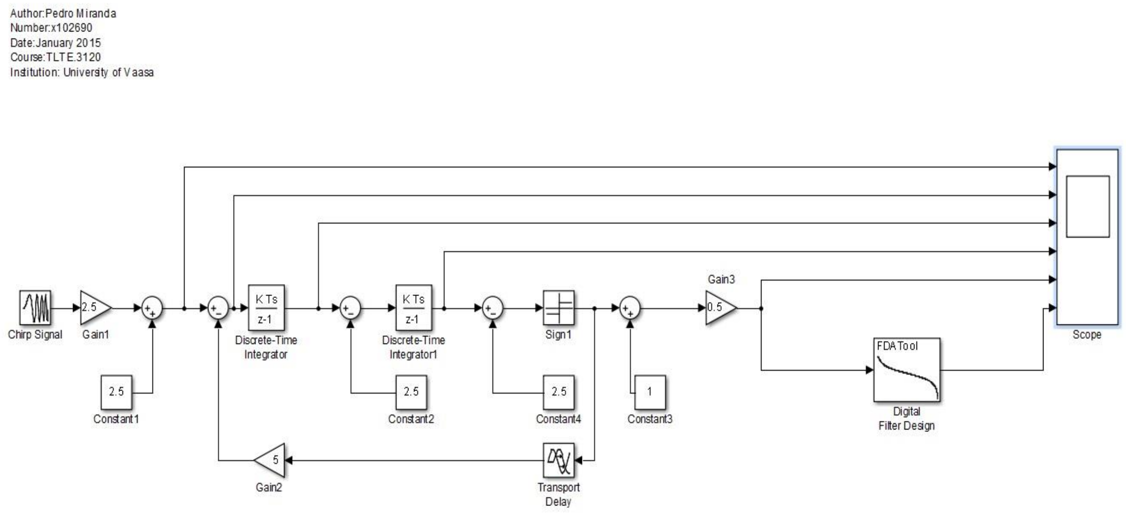

Author:Pedro Miranda
Number:x102690
Date:January 2015
Course:TLTE.3120
Institution: University of Vaasa

fig. 8 – ADC with second order delta-sigma modulation model

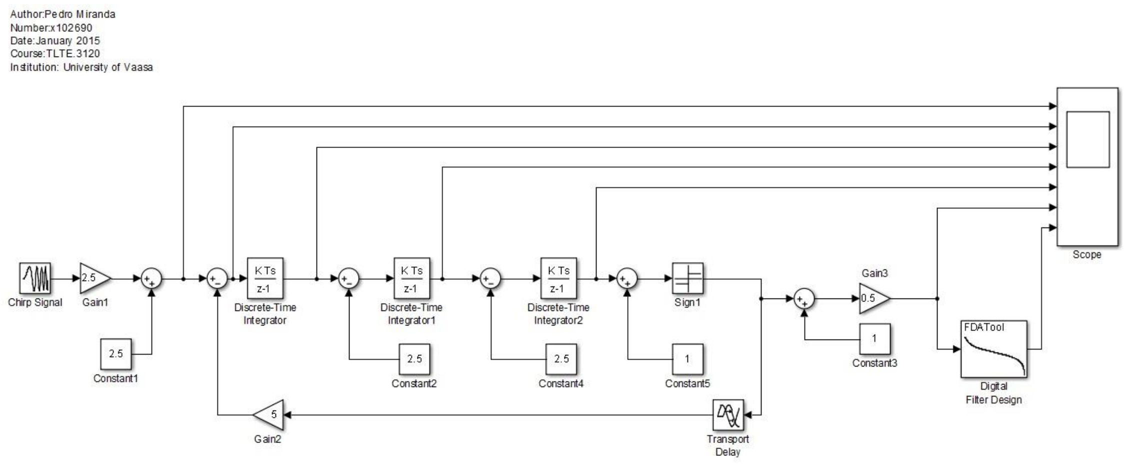


fig. 9 – ADC with third order delta-sigma model

The output of this models were the following ones:

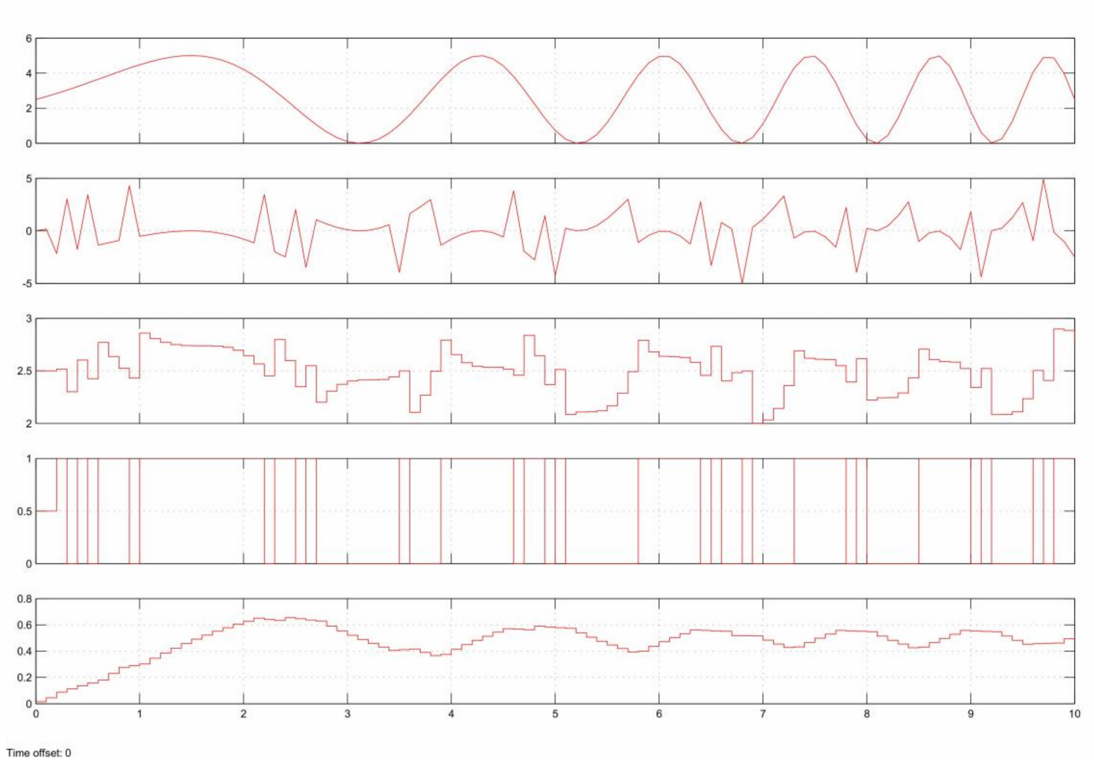


fig. 10 - output of ADC with first order delta-sigma modulation

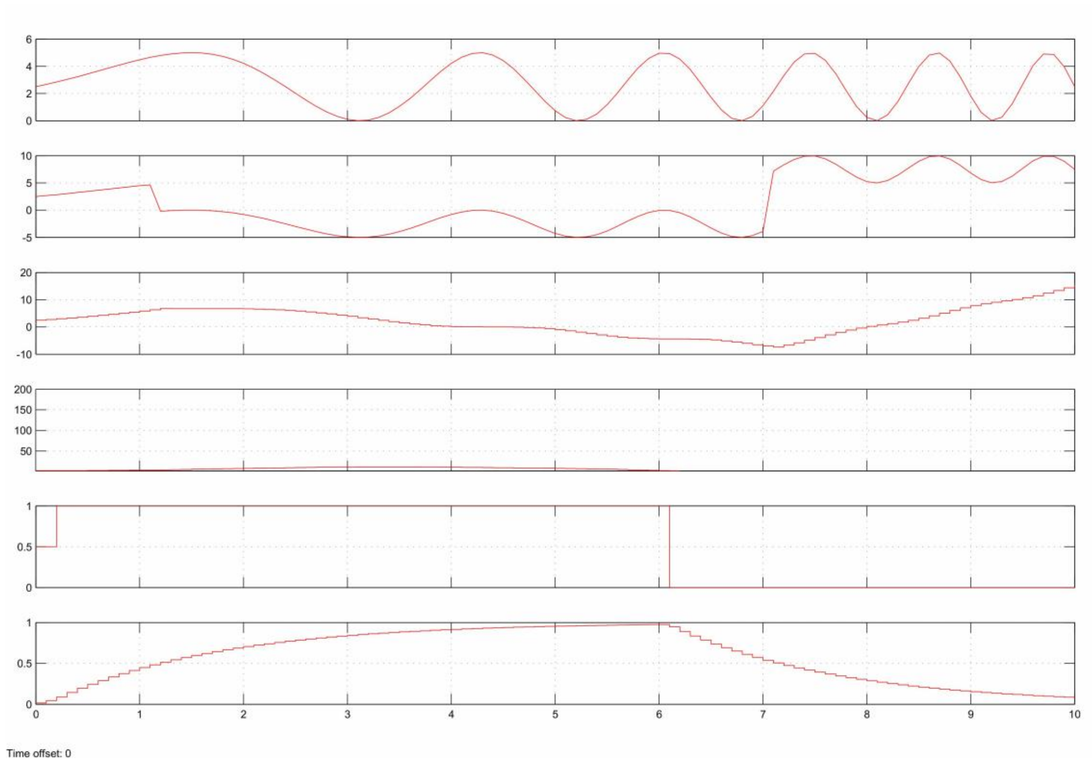


fig. 11 - output of ADC second order delta-sigma modulation

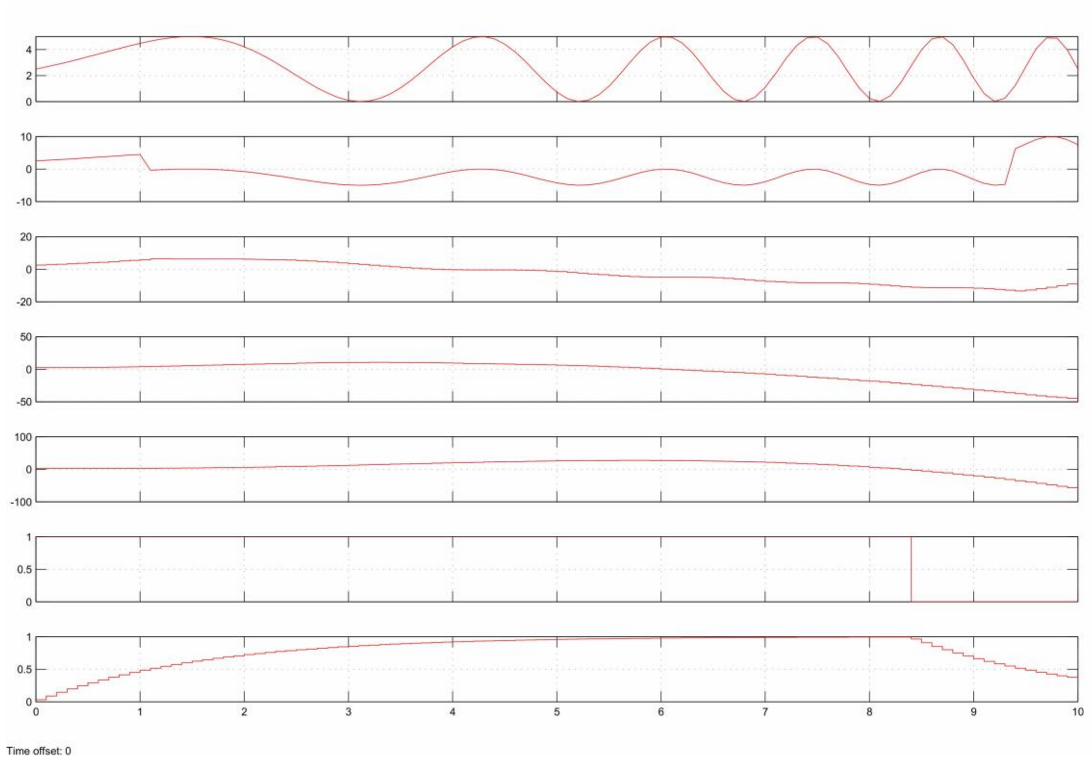


fig. 12 - output of ADC with third order delta-sigma modulation

3.3. Final results and outputs

It is possible to understand from the reading of the outputs in fig. 10, fig. 11 and fig. 12 that the analog signal converted to digital signal was not correctly converted.

This fact it was caused for lack of knowledge of myself in the signals area.

It was impossible for me to understand – in time – the way to manipulate the signals with FDA tool and with the other procedures that Simulink permits.

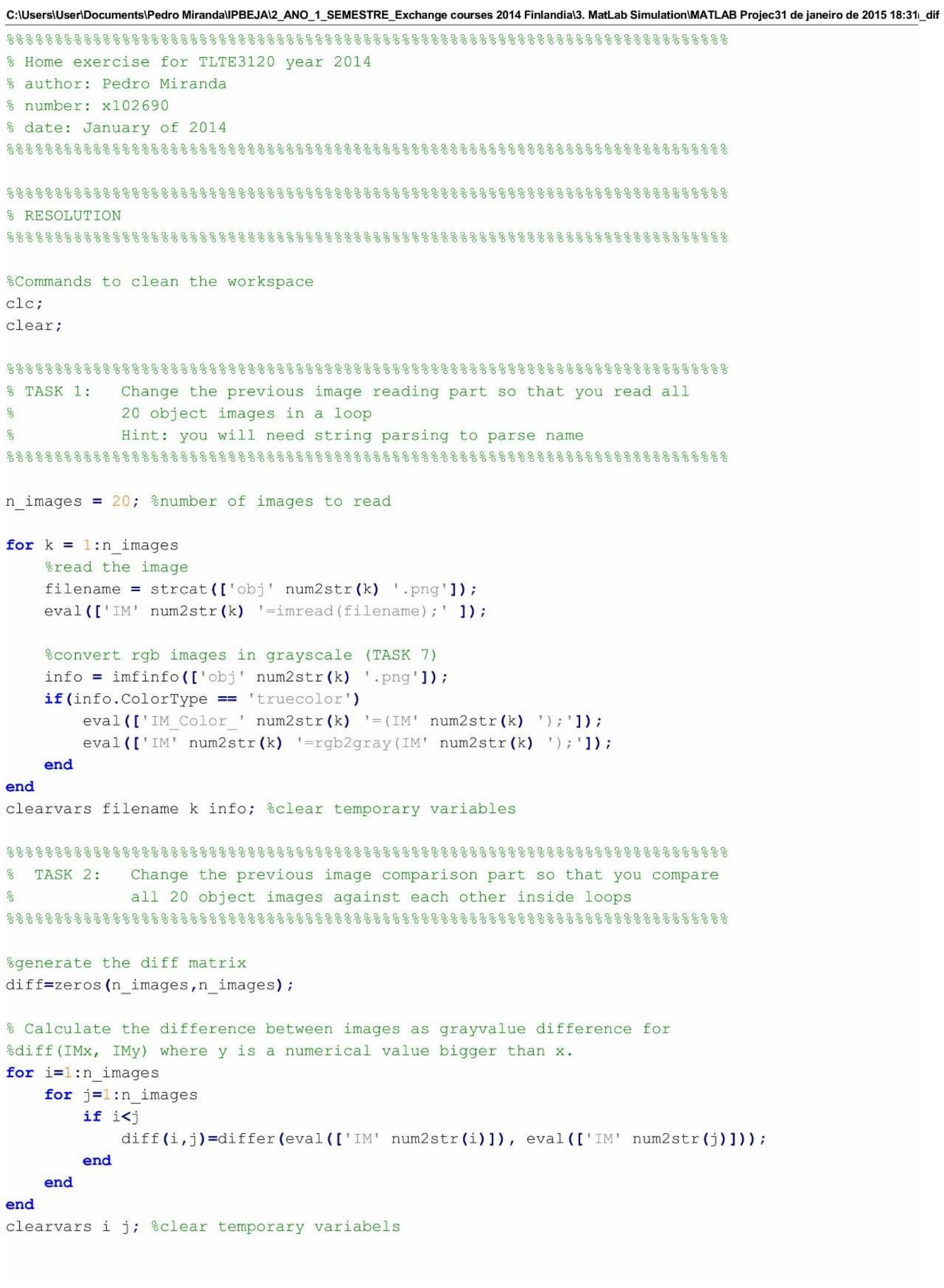
4. Conclusion

However this homework does not present perfect solutions, it was extremely important in the consolidation of knowledge acquired during the course.

Matlab is a powerful tool – with lots of applications in many fields. During this course were given small steps that will permit to the interested student to know how to explore Matlab in his field of investigation.

5. Attachments

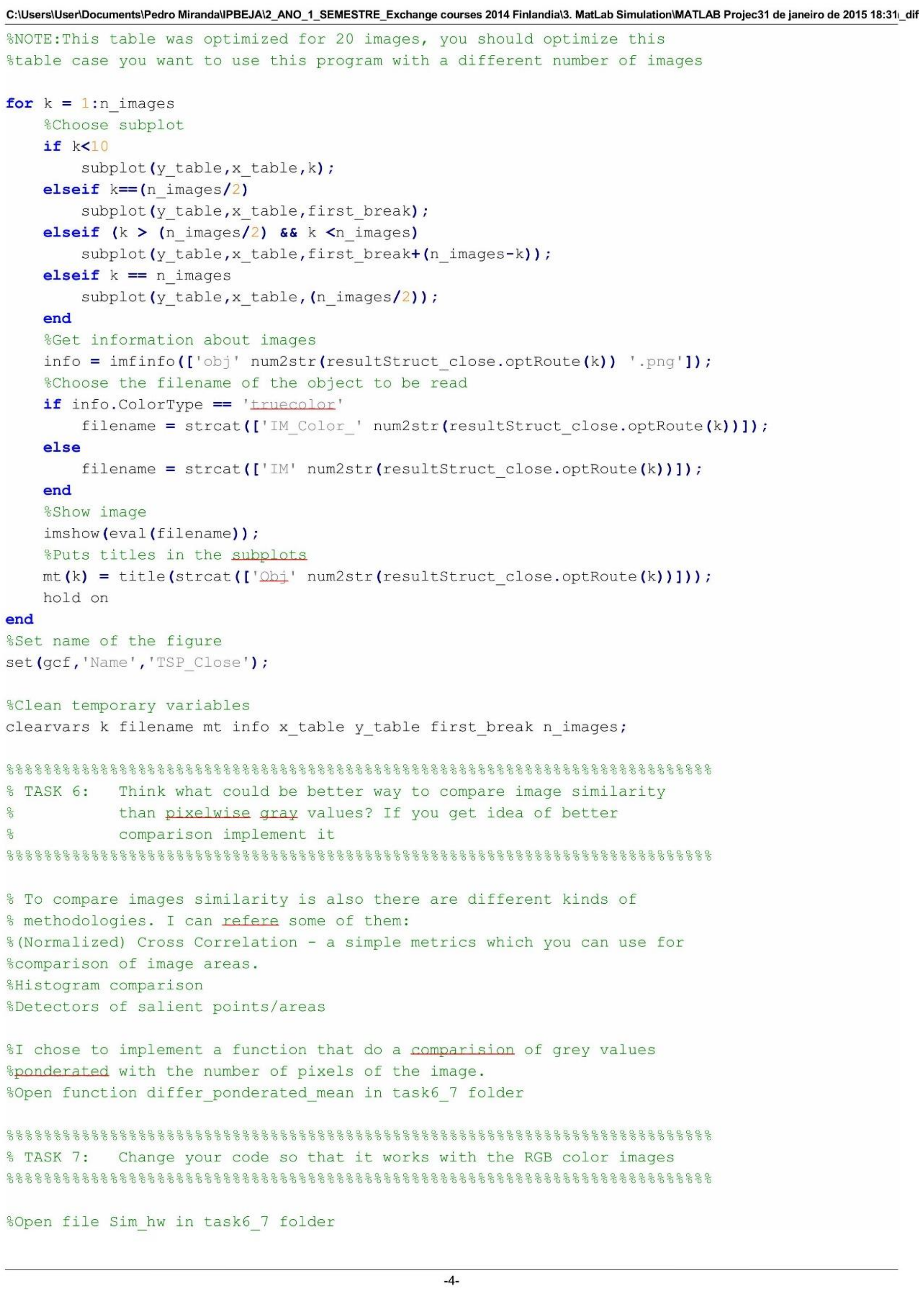
Programming task code



C:\Users\User\Documents\Pedro Miranda\IPBEJA\2_ANO_1_SEMESTRE_Exchange courses 2014 Finlandia\3. MatLab Simulation\MATLAB Projec31 de janeiro de 2015 18:31_dif

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Home exercise for TLTE3120 year 2014
% author: Pedro Miranda
% number: x102690
% date: January of 2014
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% RESOLUTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Commands to clean the workspace
clc;
clear;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TASK 1:   Change the previous image reading part so that you read all
%           20 object images in a loop
%           Hint: you will need string parsing to parse name
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

n_images = 20; %number of images to read

for k = 1:n_images
    %read the image
    filename = strcat(['obj' num2str(k) '.png']);
    eval(['IM' num2str(k) '=imread(filename);' ]);

    %convert rgb images in grayscale (TASK 7)
    info = imfinfo(['obj' num2str(k) '.png']);
    if(info.ColorType == 'truecolor')
        eval(['IM_Color_' num2str(k) '=(IM' num2str(k) ');']);
        eval(['IM' num2str(k) '=rgb2gray(IM' num2str(k) ');']);
    end
end
clearvars filename k info; %clear temporary variables

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  TASK 2:   Change the previous image comparison part so that you compare
%            all 20 object images against each other inside loops
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%generate the diff matrix
diff=zeros(n_images,n_images);

% Calculate the difference between images as grayvalue difference for
%diff(IMx, IMy) where y is a numerical value bigger than x.
for i=1:n_images
    for j=1:n_images
        if i<j
            diff(i,j)=differ(eval(['IM' num2str(i)]), eval(['IM' num2str(j)]));
        end
    end
end
clearvars i j; %clear temporary variabels
```

```matlab
clearvars userConfig %clear temp variabels

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TASK 5:   Plot the two shortest routes with subimage method similarly
%           as shown in the example
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Display TSP_open
figure

%Dimensions of the plot
x_table = 9;
y_table = 4;

%NOTE:This table was optimized for 20 images, you should optimize this
%table case you want to use this program with a different number of images

for k = 1:n_images
    %Chosing the subplot
    if k <  n_images/2
        subplot(y_table,x_table,k);
    elseif k == n_images/2
        subplot(y_table,x_table,18);
    elseif k == n_images/2 + 1
        subplot(y_table,x_table,27);
    elseif k > n_images/2 + 1
        subplot(y_table,x_table,(y_table * x_table)-(k-(n_images/2 + 2)));
    end
    %Get information about images
    info = imfinfo(['obj' num2str(resultStruct_open.optRoute(k)) '.png']);
    %Choose the filename of the object to be read
    if info.ColorType == 'truecolor'
        filename = strcat(['IM_Color_' num2str(resultStruct_open.optRoute(k))]);
    else
        filename = strcat(['IM' num2str(resultStruct_open.optRoute(k))]);
    end
    %Show image
    imshow(eval(filename));
    %Puts titles in the subplots
    mt(k) = title(strcat(['Obj' num2str(resultStruct_open.optRoute(k))]));
    hold on
end
%Set name of the figure
set(gcf,'Name','TSP_Open');

%Clean temporary variables
clearvars k filename mt info x_table y_table;


%Display TSP_open
figure

%Dimensions of the plot
x_table = 9;
y_table = 3;
first_break = x_table * (y_table - 1);
```

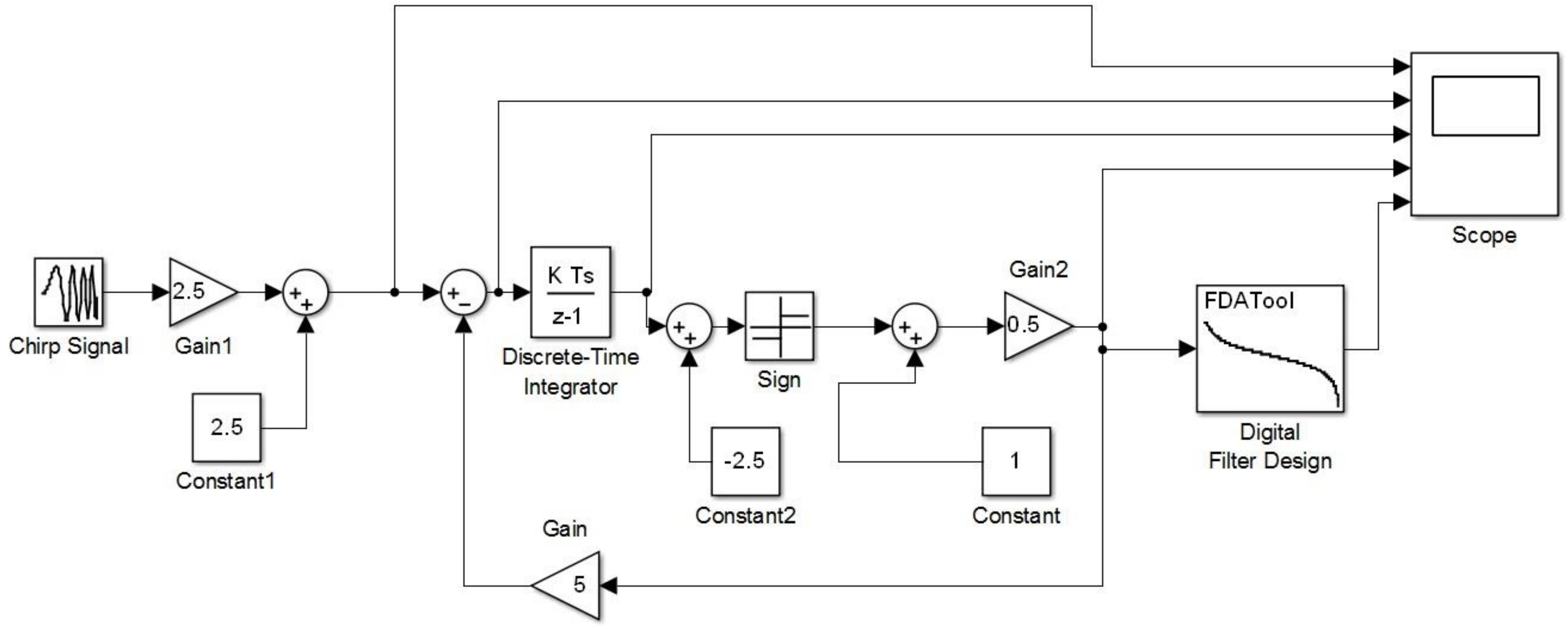


C:\Users\User\Documents\Pedro Miranda\IPBEJA\2_ANO_1_SEMESTRE_Exchange courses 2014 Finlandia\3. MatLab Simulation\MATLAB Projec31 de janeiro de 2015 18:31_dif

```
%NOTE:This table was optimized for 20 images, you should optimize this
%table case you want to use this program with a different number of images

for k = 1:n_images
    %Choose subplot
    if k<10
        subplot(y_table,x_table,k);
    elseif k==(n_images/2)
        subplot(y_table,x_table,first_break);
    elseif (k > (n_images/2) && k <n_images)
        subplot(y_table,x_table,first_break+(n_images-k));
    elseif k == n_images
        subplot(y_table,x_table,(n_images/2));
    end
    %Get information about images
    info = imfinfo(['obj' num2str(resultStruct_close.optRoute(k)) '.png']);
    %Choose the filename of the object to be read
    if info.ColorType == 'truecolor'
        filename = strcat(['IM_Color_' num2str(resultStruct_close.optRoute(k))]);
    else
        filename = strcat(['IM' num2str(resultStruct_close.optRoute(k))]);
    end
    %Show image
    imshow(eval(filename));
    %Puts titles in the subplots
    mt(k) = title(strcat(['Obj' num2str(resultStruct_close.optRoute(k))]));
    hold on
end
%Set name of the figure
set(gcf,'Name','TSP_Close');

%Clean temporary variables
clearvars k filename mt info x_table y_table first_break n_images;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TASK 6:   Think what could be better way to compare image similarity
%           than pixelwise gray values? If you get idea of better
%           comparison implement it
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% To compare images similarity is also there are different kinds of
% methodologies. I can refere some of them:
%(Normalized) Cross Correlation - a simple metrics which you can use for
%comparison of image areas.
%Histogram comparison
%Detectors of salient points/areas

%I chose to implement a function that do a comparision of grey values
%ponderated with the number of pixels of the image.
%Open function differ_ponderated_mean in task6_7 folder

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TASK 7:   Change your code so that it works with the RGB color images
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Open file Sim_hw in task6_7 folder
```

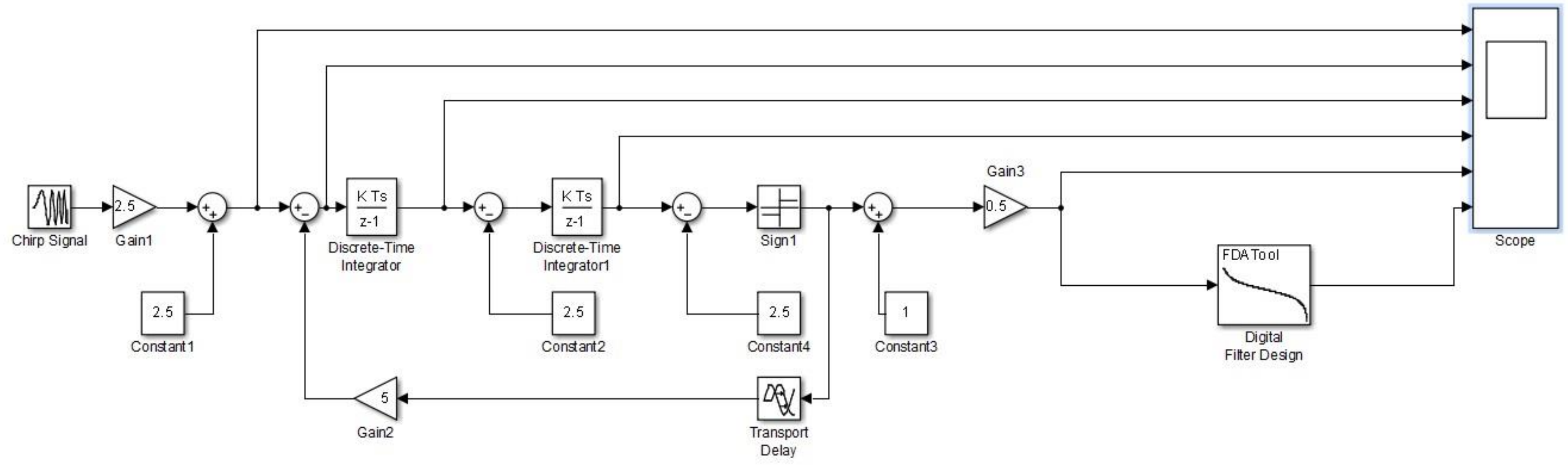
Simulink models and outputs

ADC with first order delta-sigma modulation model

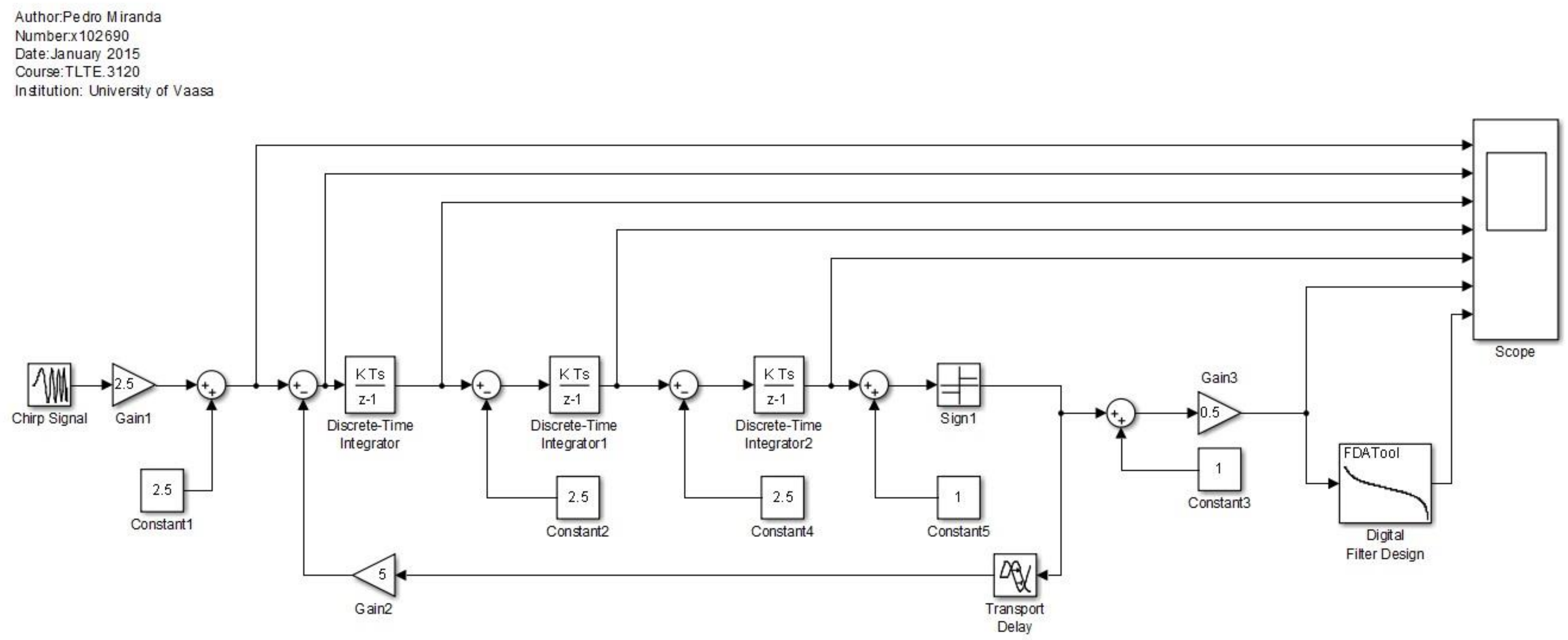


ADC with second order delta-sigma modulation model

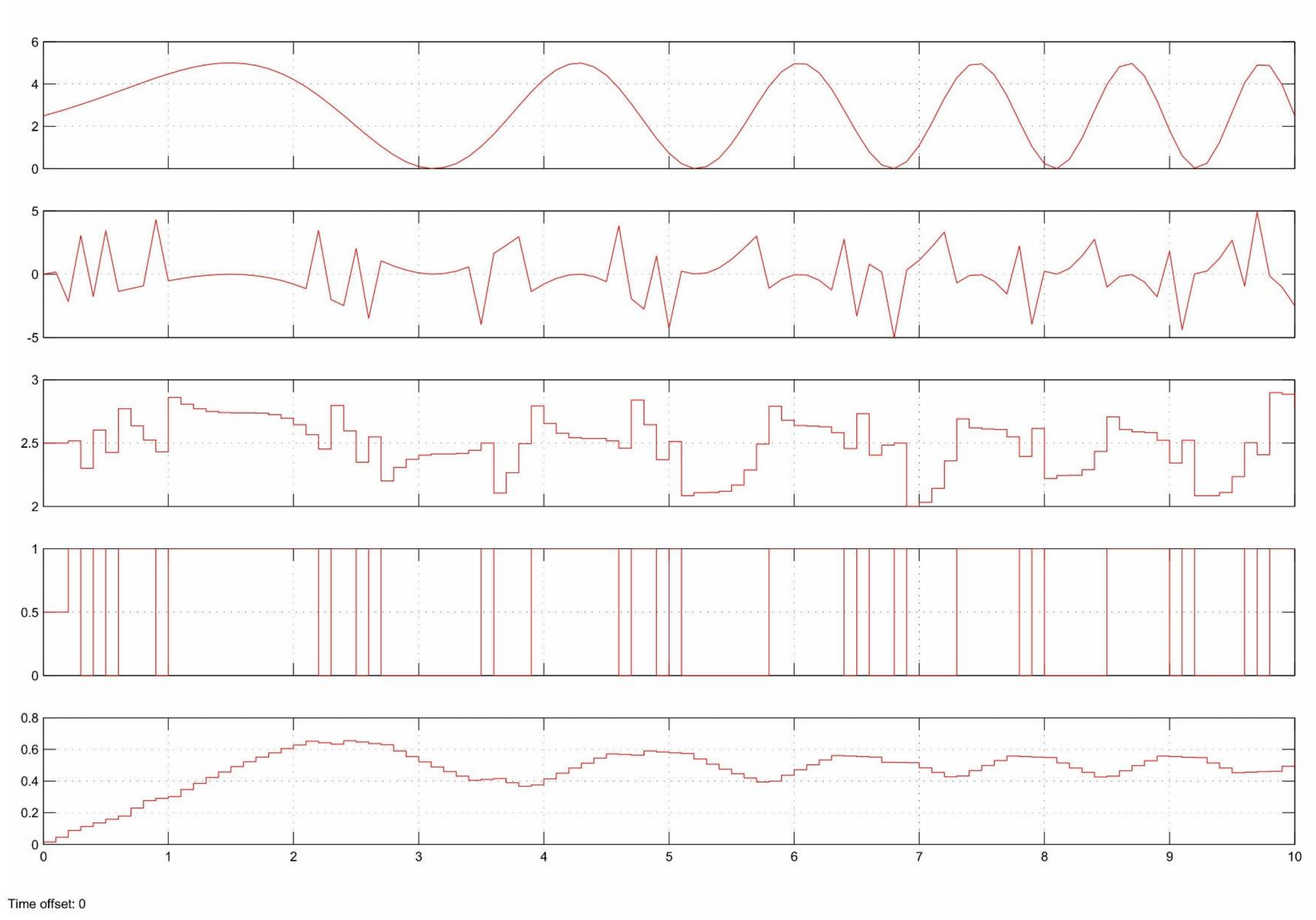
Author:Pedro Miranda
Number:x102690
Date:January 2015
Course:TLTE.3120
Institution: University of Vaasa



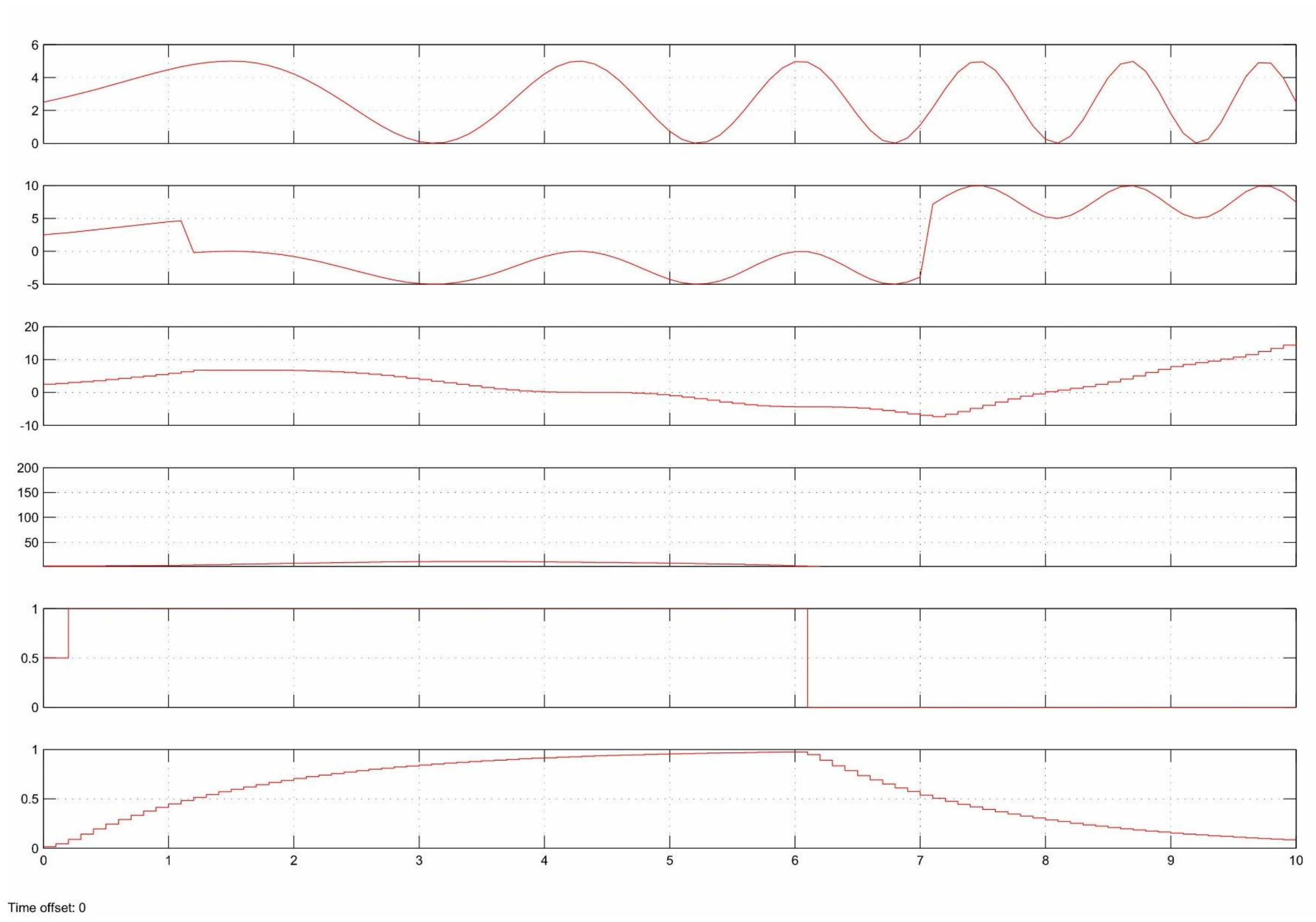
ADC with third order delta-sigma modulation model



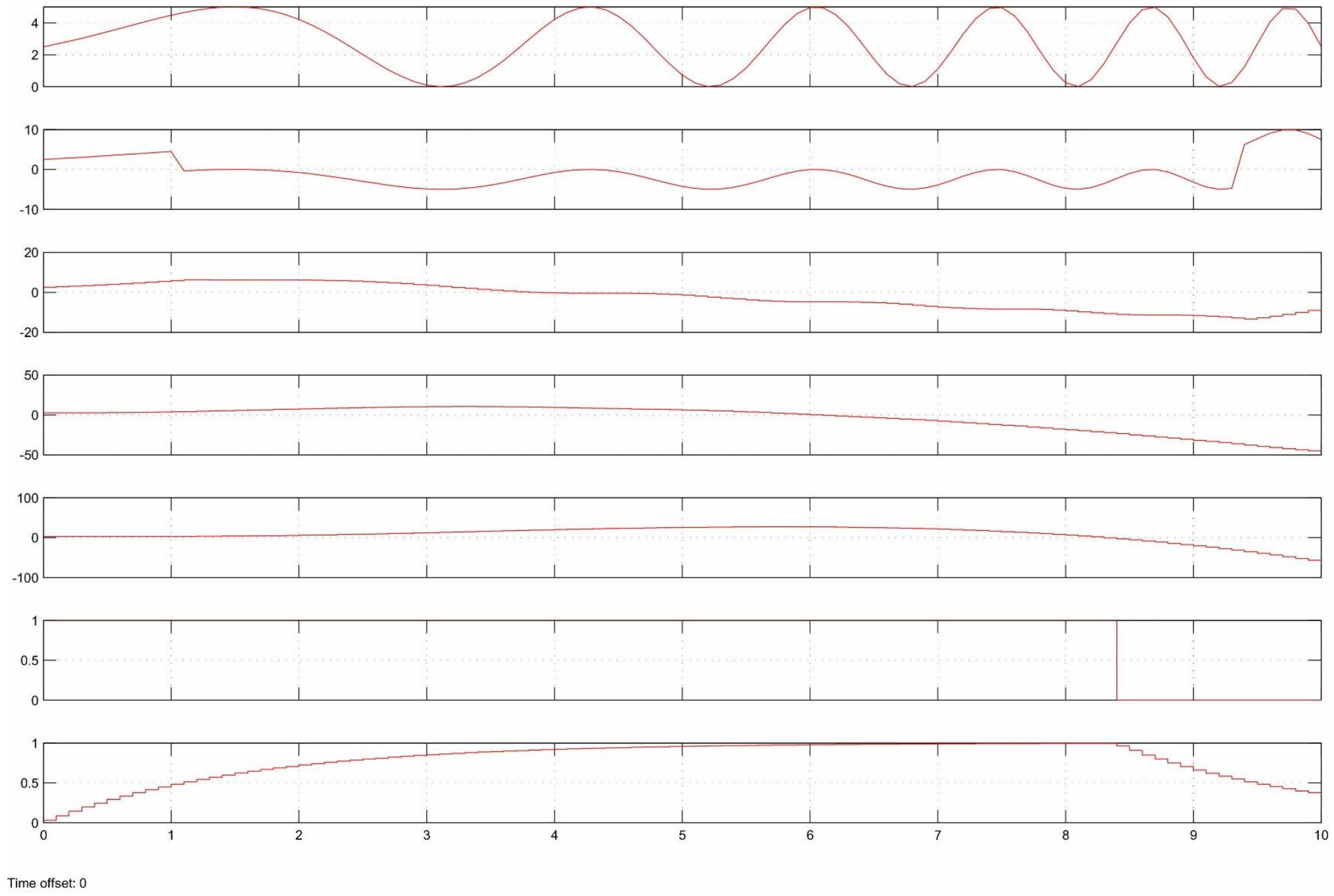
Output of ADC first order delta-sigma modulation



Output of ADC second order delta-sigma modulation



Output of ADC third order delta-sigma modulation


Time offset: 0