Exercise 1: Estimation

LEDS Exam: September 11, 2023

Then, the identification problem should be solved following the steps below:

1. **Understand the model structure**: The provided signals refer to either a ARX or an AR model. You should get the structure straight away from the measurements. Then display a coordinate graph where the x coordinate is the input signal and and the y coordinate is the output signal (something like plot(x,y,'o')), each coordinate couple (u(t),y(t)) should correspond to a point in the plot. The result is different if you do it for an AR o an ARX process.
2. **Define the required estimation algorithm:** describe the algorithm you plan to use and why and add brief description of the matlab function you implemented accordingly. In this particular case you are asked to implement the batch version of the estimation algorithm.
3. **Estimate the model order:** describe the criteria you plan to use and add brief description of the matlab function you implemented to do that.
4. **Validate the model:** describe the criteria with hypothesis testing you plan to use to get model validation and add a description of the matlab functions you implemented to do that.

```
clear all;
close all;
addpath ./Functions;
```

1] System measurements

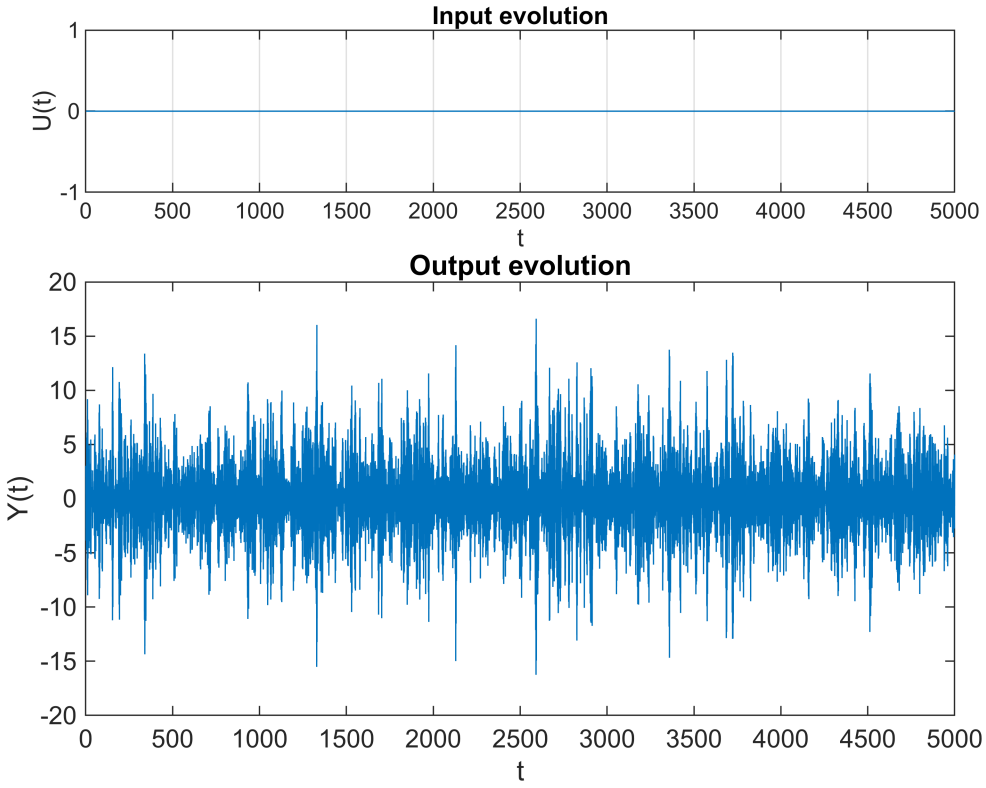
```
N=5000;
Student = 'Simone Cenerini';
Matriculation = '0000983677';
[Measurements] = IdentifyThis(N, Student, Matriculation);
u=Measurements.u;
y=Measurements.y;
```

2] Data analysis & Understand the model structure:

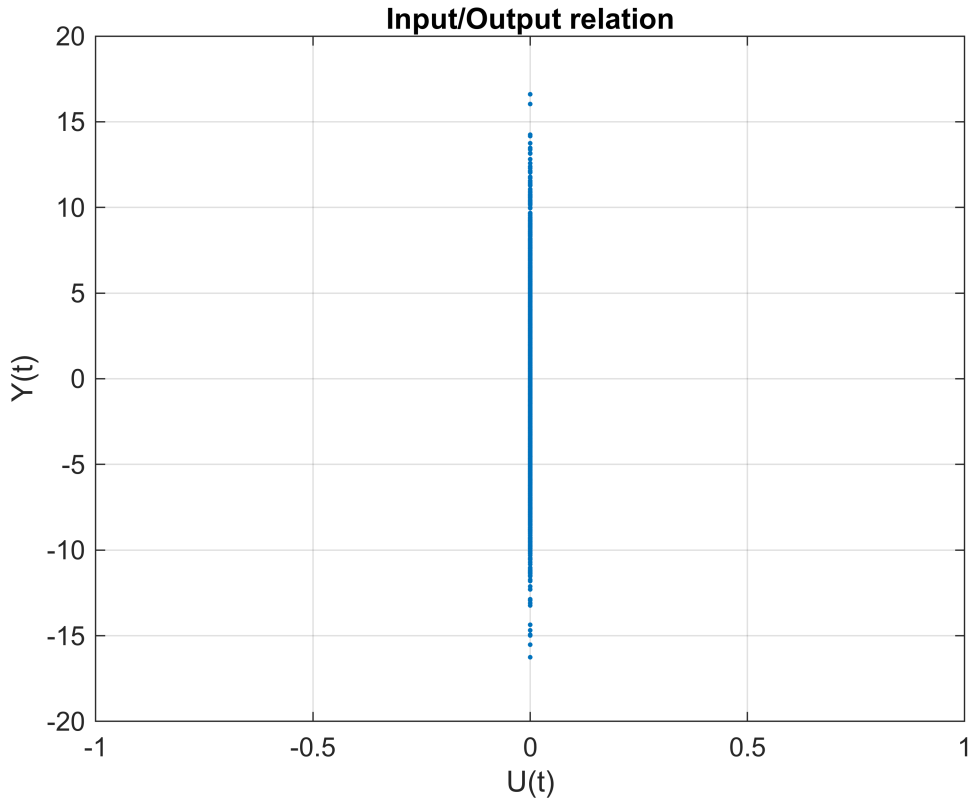
We plot our obtained measurements:

```
figure();
subplot(3,1,1);
plot(u);
grid on;
ylabel('U(t)');
xlabel('t');
title('Input evolution');

subplot(3,1,[2,3]);
plot(y);
xlim([0 N]);
ylabel('Y(t)');
xlabel('t');
```

```
title('Output evolution');
```




```
figure();
plot(u,y,'.');
grid on;
ylabel('Y(t)');
xlabel('U(t)');
title('Input/Output relation');
```



As we can see the input samples are identically equal to zero, this means that we don't have the input in our system and we can conclude that our system can be represented by a time series model, a model without the input, a *driving noise process*.

In particular we can conclude that we are dealing with an **Auto-Regressive Time Series Model**.

The output is a typical stochastic output of a process.

The general equation of an **AR Model** of *order n* is the following:

$$y(t) + a_1 y(t-1) + \ldots + a_n y(t-n) = w(t)$$

where $w(t)$ is a *white process* with 0 mean and variance σ_w^2.

We can write our problem according to the linear regression form:

$$y(t) = \varphi^T(t)\theta + e(t)$$

where the regressor has the following expression $\varphi(t) = [-y(t-1) \cdots - y(t-n)]^T = \left[\; -\varphi_y^T(t) \; \right]$.

Our goal is to estimate the parameters vector $\theta = [a_1 \cdots a_n]^T$.

We will put ourserlves in the persepective of an observer that is able only to measure the output of this AR process and does not know the value of n (the model order) and the first of our problem is to retrieve it (chapter 4).

3] Define the required estimation algorithm:

Since we have all the data available during the estimation, we were asked to use a batch version algorithm to estimate our parameters. In particular we are going to use a Least Square estimator, based on the minimization of the following loss function:

$$J(\theta) = \frac{1}{N}\sum_{t=1}^{N}\left(y(t) - \varphi^T(t)\,\theta\right)^2$$

Available in *"MyCostFunc.m"*

By computing the derivatives of $J(\theta)$, in order to find a local minimum, is possible to observe that the loss function is quadratic. According to this quadratic nature of the loss function we can obtain a closed-form solution for the least square estimator:

$$\hat{\theta}_{\mathrm{LS}} = \left(\frac{H^T H}{N}\right)^{-1}\left(\frac{H^T Y}{N}\right)$$

Available in *"MyLS.m"*

This formula requires the knowledge of the Hankel matrix H and the obsterved outputs Y. Since we are dealing with an AR model we can say that H has the following expression:

$$H = \begin{bmatrix} -y(0) & -y(-1) & \cdots & -y(1-n) \\ \vdots & \vdots & \ddots & \vdots \\ -y(N-1) & -y(N-2) & \cdots & -y(N-n) \end{bmatrix}$$

Available in *"MyHankel.m"*

It's important to notice that if the disturbance was colored (for example an ARMAX process) instead of the white one, our cost function will not anymore be quadratic and so the optimal value for the parameter θ has to be computed in a recursive way, addressing the Prediction Error Method.

We can also highlight that there exist also a Recursive version of the Least Square algorithm that can be used in order to increase the performances in speed sense and case we are dealing with a real time system.

4] Estimate the model order:

We are going to find the possible order n from 0 to PP value. (Please note that in AR model $n \equiv p$)

```
PP = 10;
```

4.1] Loss function analisys

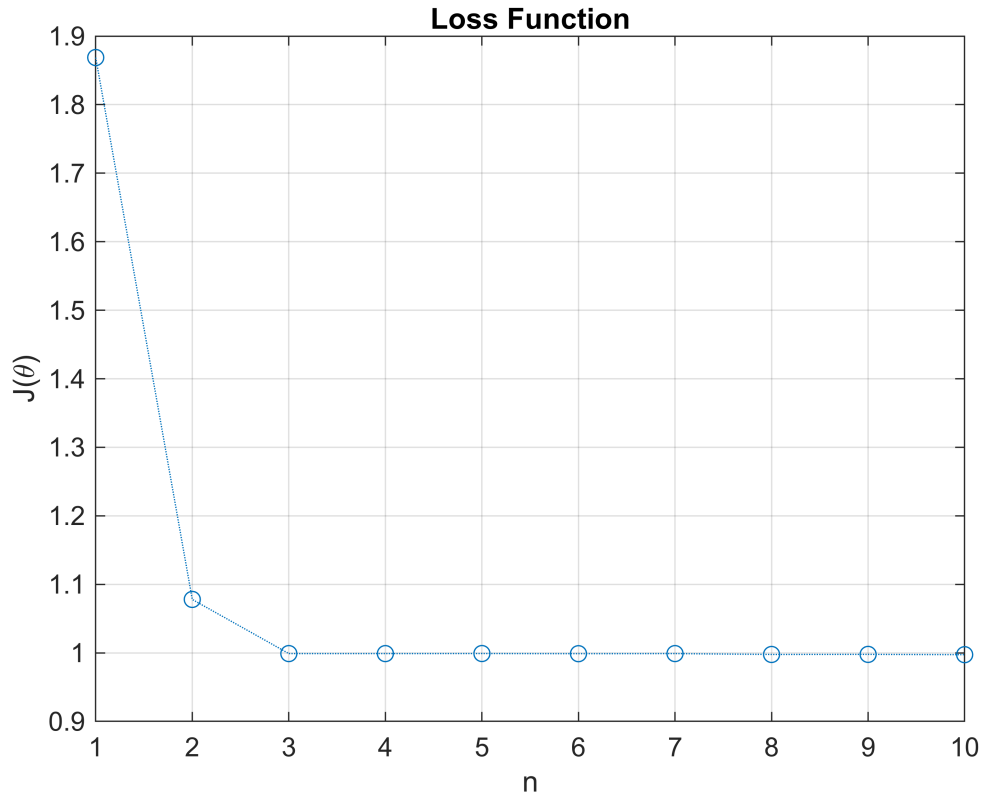
Evaluate the cost function until the order 10

```
J=zeros(PP,1);

for n=1:PP
    Hy = MyHankel(y,n);
    H=[-Hy];
```

```
    Theta_LS = MyLS(y,n,H);
    J(n) = MyCostFunc(y,H,Theta_LS,n);
end
```

Look at its plot

```
figure();
plot(J,':o');
grid on;
ylabel('J(\theta)');
xlabel('n');
title('Loss Function');
```



We can note that the loss function decreases faster until order 3 with respect to higher orders.

So, we can presume that the order of our system could be n=3.

4.2] Cost function estimation - $J(\theta)$

The first method we try is to split the training set in two parts and so to obtain a *training set* and a *validation set.* Then:

1. Find the estimate $(\hat{\theta}_{\text{ls}})$ by using only the first half of the data set (*training set)*
2. Plot the cost function behaviour $(J(\hat{\theta}_{\text{ls}}, Y_{\text{vl}}))$ by using the second half of the data set (*validation set)*

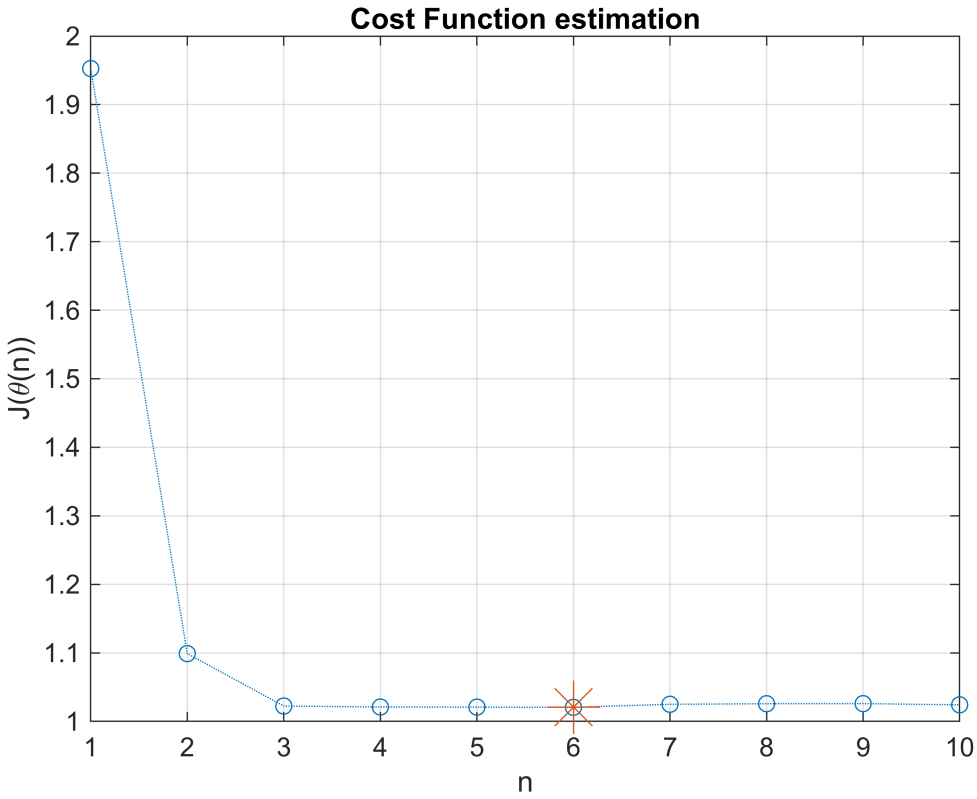
```
y_tr = y(1:N/2);
y_vl = y(N/2+1:end);

J_theta=zeros(PP,1);

for n=1:PP
    Hy = MyHankel(y_tr,n);
    H=[-Hy];
    Theta_LS = MyLS(y_tr,n,H);
    Hy = MyHankel(y_vl,n);
    H=[-Hy];
    J_theta(n) = MyCostFunc(y_vl,H,Theta_LS,n);
end
```

Now plot the result

```
[Theta_J,n_J]=min(J_theta); %Find minimun
fprintf('The minimun value is given with the order %d', n_J);
```

```
The minimun value is given with the order 6
```

```
figure();
plot(J_theta,':o');
grid on;
hold on;
plot(n_J,J_theta(n_J),'*','MarkerSize',20); %Plot minimun
ylabel('J(\theta(n))');
xlabel('n');
title('Cost Function estimation');
```



We have obtained an estimation of the model complexity by looking at the minimum of the loss function but in the most of the cases this method overestimates the complexity.

4.3] CRITERIA WITH COMPLEXITY TERM - [FPE, AIC, MDL]

Now we try to give a better solution of our *complexity selection problem* by means of the criteria with complexity terms.

These criteria are characterized by a term called 'Penalizing term' that increase the value of $J(\hat{\theta}_N)$ when the p value (model complexity) increases. This avoids overfitting and fits well the parcimony principle.

General form

$$V(\hat{\theta}_N) = N \cdot \log \mathrm{J}(\hat{\theta}_N) + f(N, p)$$

FPE (Final Prediction Error)

$$FPE = J(\hat{\theta}_N) - \frac{2p}{N} J(\hat{\theta}_N)$$

AIC (Akaike Information Criterion)

$$AIC = N \cdot \log \mathrm{J}(\hat{\theta}_N) + 2p$$

MDL (Minimum Description Length)

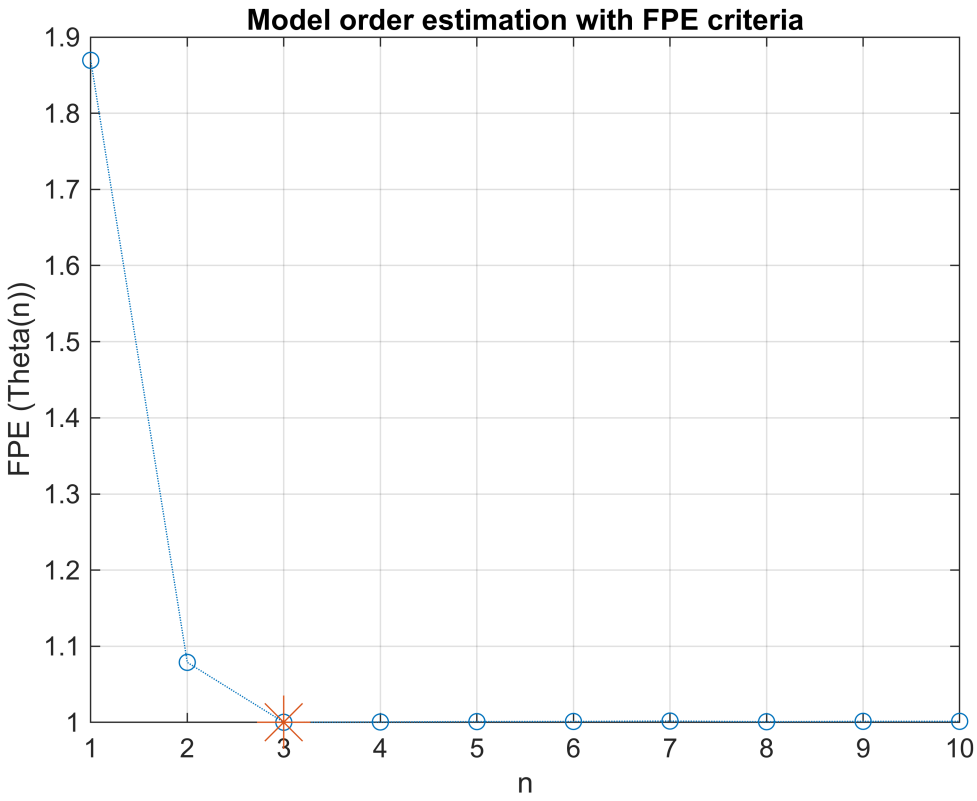
$$MDL = N \cdot \log \mathrm{J}\left(\widehat{\theta_N}\right) + 2p \cdot logN$$

All these methods are going to be implemented by using the training set, so we are going to reuse all the data that we have.

```
FPE = zeros(PP,1);
AIC = zeros(PP,1);
MDL = zeros(PP,1);

for n=1:PP
    H=-MyHankel(y,n);
    Theta_LS = MyLS(y,n,H);
    J_n = MyCostFunc(y,H,Theta_LS,n);

    FPE(n) = J_n * ((N+n)/(N-n));
    AIC(n) = N*log(J_n) + 2*n;
    MDL(n) = N*log(J_n) + 2*n*log(N);
end
```

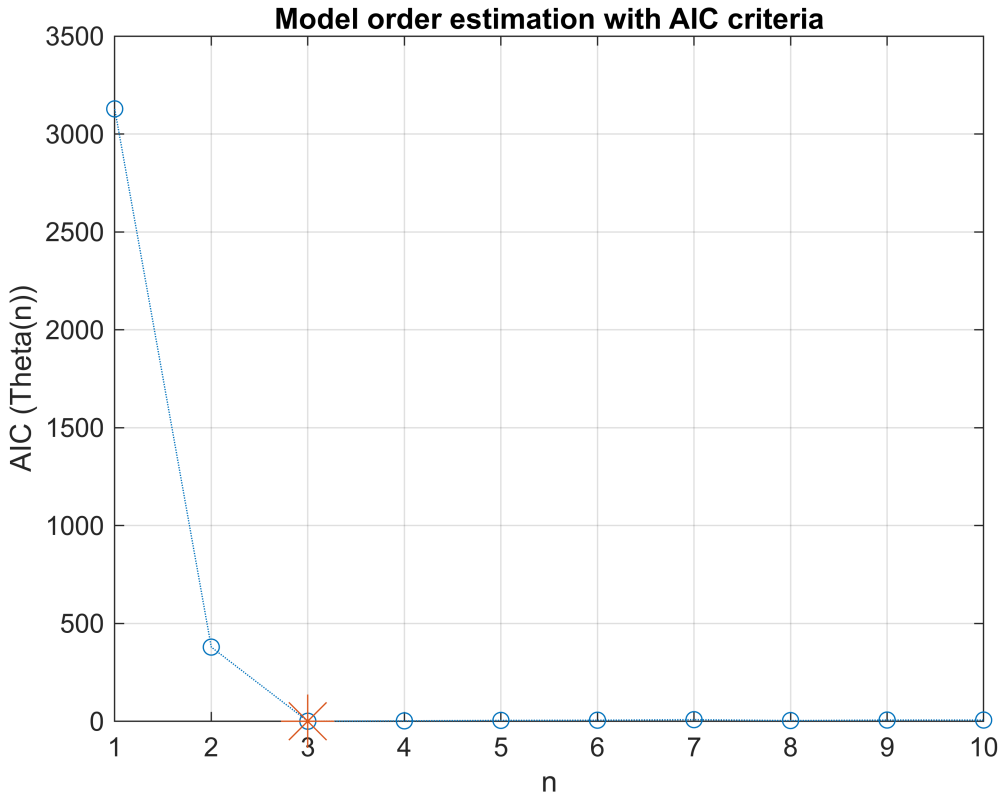
Now we are going to plot the results obtained from each criteria

```
[~, n_FPE]=min(FPE);
Plot_complexity_term(FPE,n_FPE,'FPE');
```

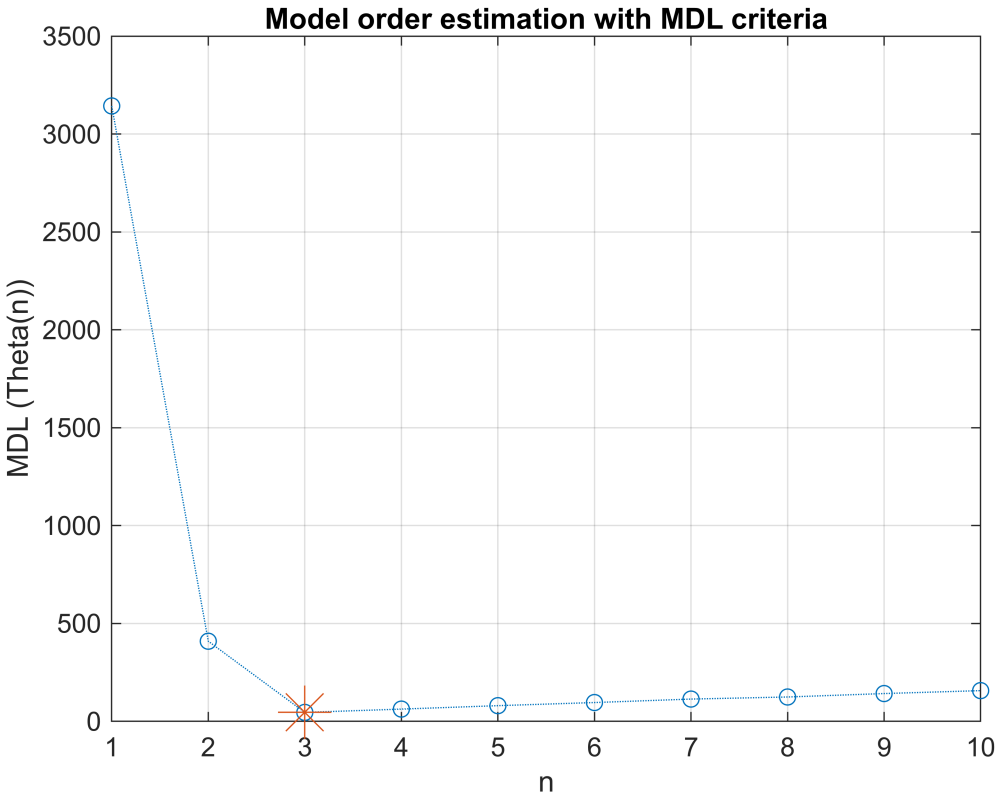



```
[~, n_AIC]=min(AIC);
```

```
Plot_complexity_term(AIC,n_AIC,'AIC');
```




```
[~, n_MDL]=min(MDL);
Plot_complexity_term(MDL,n_MDL,'MDL');
```



4.4] Model order choice:

Now we are going to choose the model order by select the MDL result since is possible to prove it is a consistent method for order estimation.

```
fprintf('Order estimated from the different critiria:\n n_J \t n_FPE \t n_AIC \t n_MDL \n %d \t
```

```
Order estimated from the different critiria:
 n_J      n_FPE      n_AIC      n_MDL
 6      3      3      3
```

After having runned severals times “IdentifyThis.p”, if output samples are every time of the same system, I will declare that system is to order 3.

```
n_star = 3;
```

5] Parameter estimation:

Once we have identified the order of the system, we are going to estimate the value of the parameter.

We are going to select different amount of samples to understend which is a reasonable value of samples.

```
N_samples = [10, 100, 1000, 5000, 10000, 100000, 1000000];
Theta_samples = zeros(length(N_samples),n_star);

for t= 1:length(N_samples)
    y = IdentifyThis(N_samples(t), Student, Matriculation).y;
```

```
    H=-MyHankel(y,n_star);
    Theta_samples(t,:) = MyLS(y,n_star,H);

    fprintf('For N = %d \nTheta_1: %.4f\tTheta_2: %.4f\tTheta_3: %.4f \n', N_samples(t),[Theta_
end
```

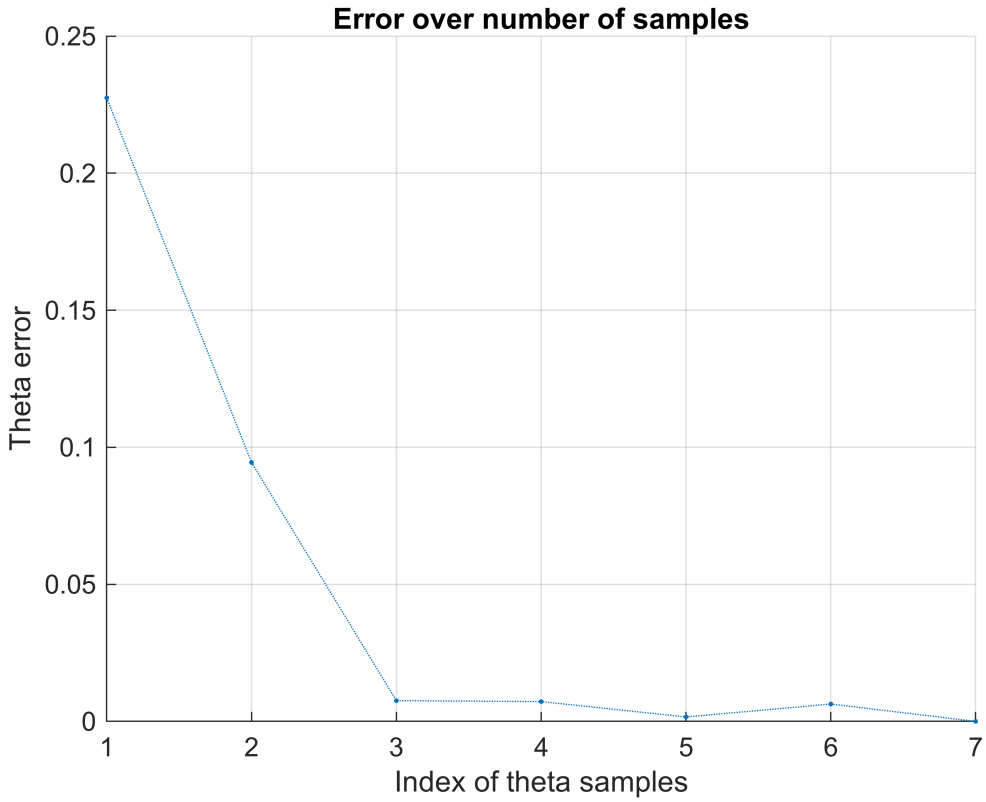
```
For N = 10
Theta_1: 1.8073    Theta_2: 1.5353    Theta_3: 0.7125
For N = 100
Theta_1: 1.7981    Theta_2: 1.2293    Theta_3: 0.3604
For N = 1000
Theta_1: 1.7215    Theta_2: 1.0499    Theta_3: 0.2549
For N = 5000
Theta_1: 1.7178    Theta_2: 1.0532    Theta_3: 0.2608
For N = 10000
Theta_1: 1.7176    Theta_2: 1.0346    Theta_3: 0.2485
For N = 100000
Theta_1: 1.7202    Theta_2: 1.0470    Theta_3: 0.2562
For N = 1000000
Theta_1: 1.7141    Theta_2: 1.0343    Theta_3: 0.2479
```

Since increase the number of samples will increase the accuracy we are going to print the error with respect to the last N used before.

```
Theta_NORM_star = norm(Theta_samples(length(N_samples),:));
Theta_errors = zeros(length(N_samples),1);
for t= 1:length(N_samples)
    Theta_errors(t) = (norm(Theta_samples(t,:))-Theta_NORM_star)/Theta_NORM_star;
end
```

Plot it:

```
figure();
grid on;
hold on;
ylabel('Theta error');
xlabel('Index of theta samples');
title('Error over number of samples');
plot(Theta_errors,':.')
```



After having runned the *"IdentifyThis.p"* several times we can conclude that a reasonable value of samples is 5000. This value is a good trade-off between a good result and the parsimony principle.

Despite this, since we have already evaluated the estimator for the bigger possible amount of data, we have chose to use this solution and not to loose the accuracy.

```
Theta_star = Theta_samples(length(N_samples),:)';
```

6] Validate the model:

In general, for the model validation, two different tests on the residuals $\varepsilon(t)$ can be used:

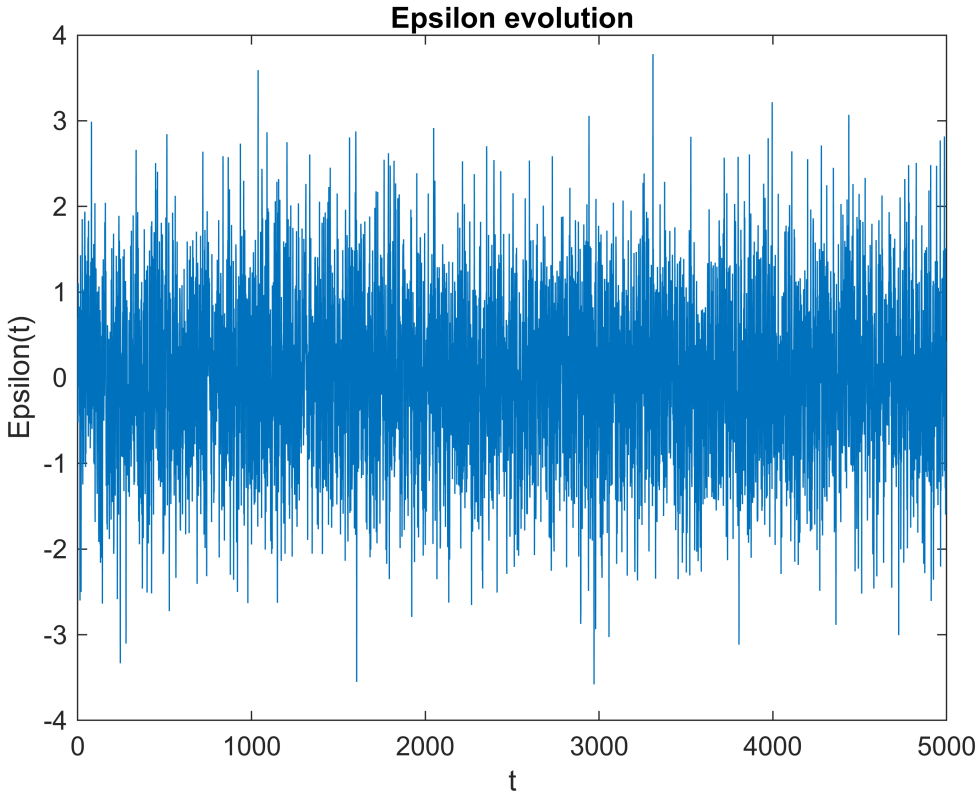
1. The whiteness test
2. The cross-correlation test between $\varepsilon(t)$ and $u(t)$

But we are dealing with an AR Model and so we can only do the whiteness test.

The **test of Cross-correlation** consists in checking if the input signal u(t) is uncorrelated with the residual ε(t). So, we want to check if the cross correlation between u(t) and ε(t) is close to 0. But in this case, we have no input, it is unapplicable.

```
% y = IdentifyThis(N, Student, Matriculation).y;
% This is non truly correct since in this way we are going to use a
% validation set and this test can be performed on the training set.
y=y(1:N); % Because in the section above we downloaded more data.
```

```
H=-MyHankel(y,n_star);
N_star = N-n_star;
Y=y(1+n_star:end);
Y_hat = H*Theta_star;
Epsilon = Y - Y_hat;

figure();
plot(Epsilon)
ylabel('Epsilon(t)');
xlabel('t');
title('Epsilon evolution');
```



6.1] Whitness test

Considering the seqence of residual ε(t) obteined from the training set.

Consider the sample variance and the first m terms of the auto correlation.

As a rule of thumb m must be chosen in the interval $m \epsilon \left[5, \frac{N}{4}\right]$.

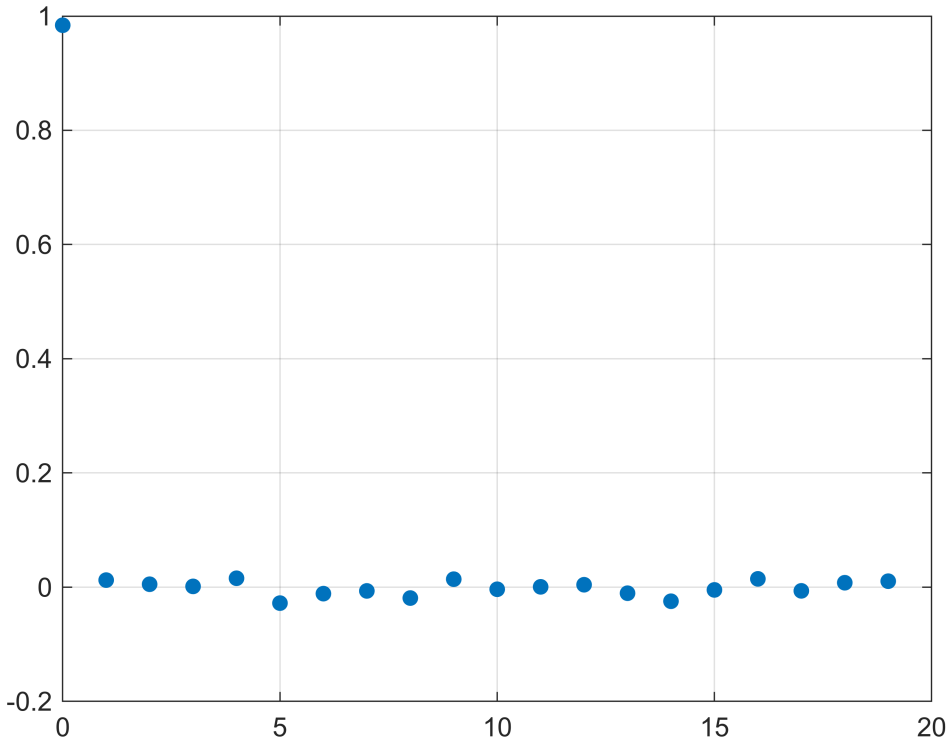
The whiteness can be seen as the following Hypotesis test:

$$\begin{cases} H_0 & \varepsilon\left(t, \widehat{\theta_N}\right) \text{ is zero mean white process} \\ H_1 & \text{not } H_0 \end{cases}$$

The function written below implements the whiteness test by computing the first m samples autocorrelations (and the sample variance):

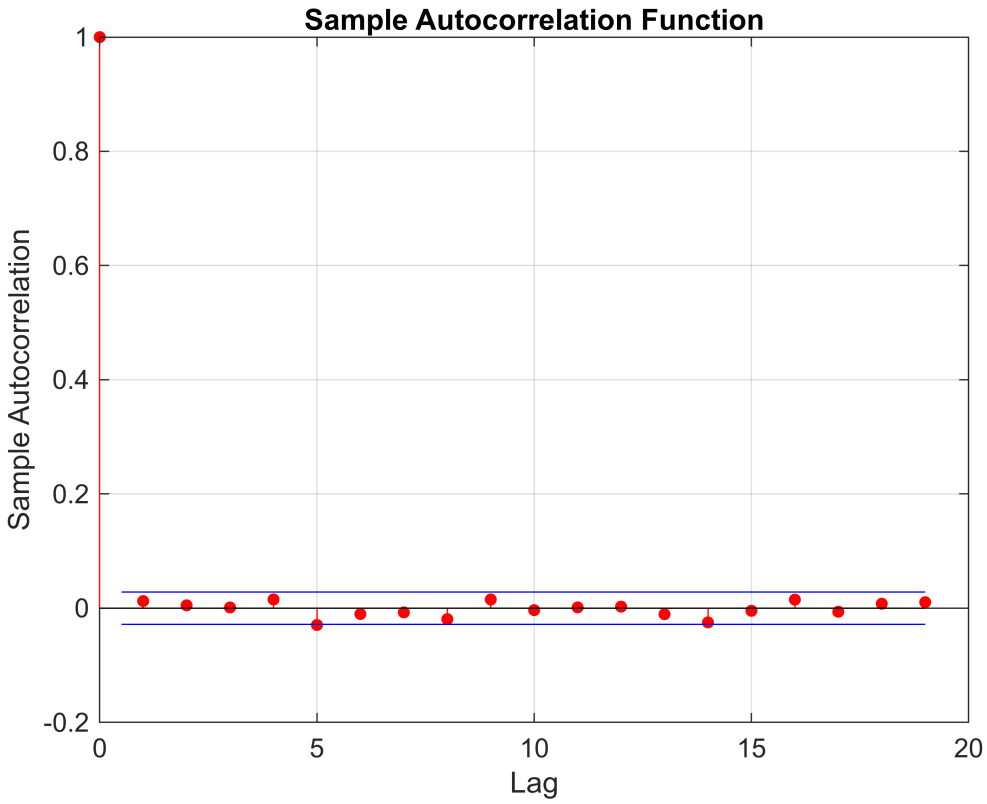
$$\hat{r}_\varepsilon = \begin{bmatrix} \hat{r}_\varepsilon(1) \\ \hat{r}_\varepsilon(2) \\ \vdots \\ \hat{r}_\varepsilon(m) \end{bmatrix}$$

```
m = 20; %from [5,N/4]
r_epsilon = zeros(m,1);
for tau = 0:m-1
    r_idx = tau+1;
    for t = 1:(N_star-tau)
        if (t-tau) > 0
            r_epsilon(r_idx)=r_epsilon(r_idx) + Epsilon(t-tau)*Epsilon(t);
        end
    end
    r_epsilon(r_idx)=r_epsilon(r_idx)*(1/N_star);
end

figure();
plot(linspace(0, m-1,m),r_epsilon,'.','MarkerSize',20);
grid on;
```



Let make it easy, try the Matlab function

```
autocorr(Epsilon,m-1);
```



We obtein the same result, good!

Just by looking the graph is possible to confirm that is a white noise.

6.1.1] Chi-square test:

Now by evaluating the following test quantity:

$$x = N\frac{\hat{r}_\varepsilon^T \hat{r}_\varepsilon}{\hat{r}_\varepsilon^2(0)} \longrightarrow \chi^2(m) \text{ when } N \to \infty$$

and check if it is Chi-Square distributed or not by implementing the following *Chi-Square distribution test*.

$$\begin{cases} H_0 : x \leq \chi_\alpha^2(m) & \varepsilon\left(t, \hat{\theta}_N\right) \text{ is zero mean white process} \\ H_1 : x > \chi_\alpha^2(m) & \text{not } H_0 \end{cases}$$

```
Alpha=0.05;
X=N*(r_epsilon(2:m)'*r_epsilon(2:m))/(r_epsilon(1)^2); %It's possible to proof that 'X' is Chi-

if X<=chi2inv(1-Alpha,m) % H0 True
     fprintf('x = %.2f <= %.2f --> Epsilon is white --> The model is valid\n', X,chi2inv(1-Alph

else % H1 True
```

```
    fprintf('x = %.2f > %.2f --> Epsilon is not white --> The model is not valid\n', X,chi2inv
end
```

```
x = 16.16 <= 31.41 --> Epsilon is white --> The model is valid
```

6.1.2] Gaussianity test:

Another possible test consists on defining the normalized quantity:

$$\widehat{\gamma}(\tau) = \frac{\widehat{r}_\varepsilon(\tau)}{\widehat{r}_\varepsilon(0)} \text{ with } \tau = 1, 2, \ldots, m$$

Now by evaluating the new test quantity:

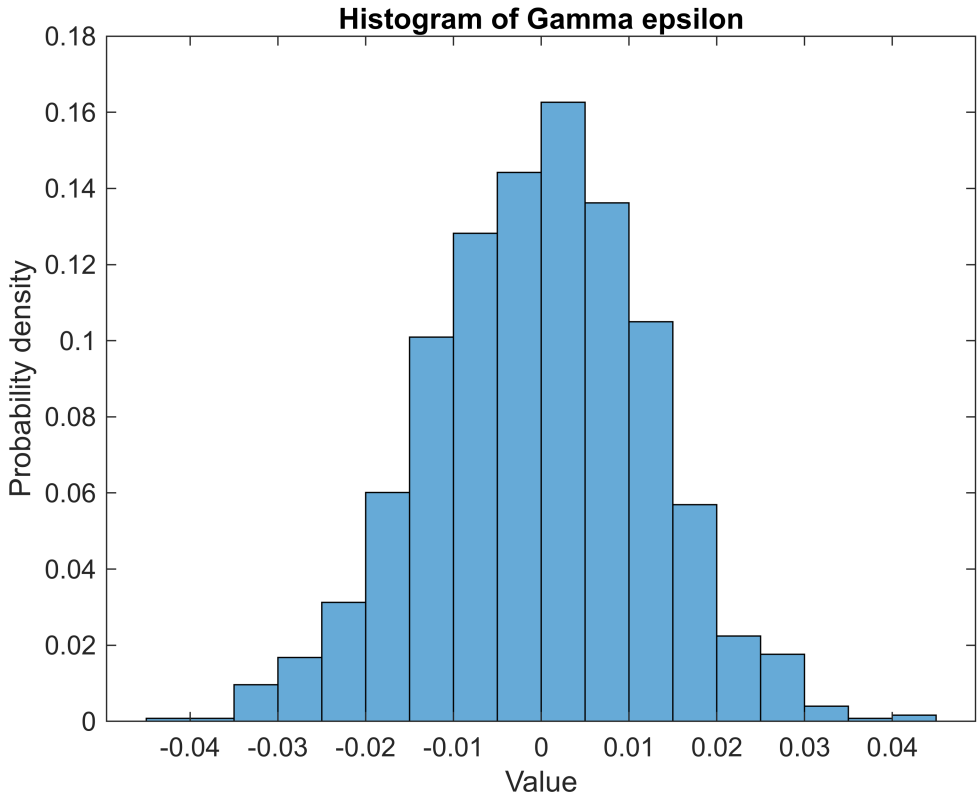
$$x = \sqrt{N}\ \widehat{\gamma}(\tau) \longrightarrow \mathcal{N}(0,1) \text{ when } N \to \infty$$

and check if it is Normal distributed or not by implementing the following Gaussianity *distribution test*.

$$\begin{cases} H_0 : |x| \leq \left(1 - \frac{\alpha}{2}\right) & H_0^\tau \\ H_1 : |x| > \left(1 - \frac{\alpha}{2}\right) & H_1^\tau \end{cases}$$

We need more samples than before, we need to perform this test for $\tau = 1, 2, \ldots, m$ after we had collected all this test we can perform the **Anderson test.**

```
m =round(N_star/4); %from [5,N/4]
r_epsilon = zeros(m,1);
for tau = 0:m-1
    r_idx = tau+1;
    for t = 1:(N_star-tau)
        if (t-tau) > 0
            r_epsilon(r_idx)=r_epsilon(r_idx) + Epsilon(t-tau)*Epsilon(t);
        end
    end
    r_epsilon(r_idx)=r_epsilon(r_idx)*(1/N_star);
end

Gamma_epsilon = zeros(m-1,1);
for tau = 1:m-1
    r_idx = tau+1;
    Gamma_epsilon(tau) = r_epsilon(r_idx)/r_epsilon(1);
end

figure();
histogram(Gamma_epsilon, 'Normalization', 'probability','BinMethod','scott') % 'Normalization'
xlabel('Value');
ylabel('Probability density');
title('Histogram of Gamma epsilon');
```



It looks like a Normal distribution.

```
Gamma = 1-(Alpha/2);
mu = 0;
sigma = 1;
H1 = 0; %Number of failed test
for tau = 1:m-1
    X=sqrt(N)*Gamma_epsilon(tau); %It's possible to proof that 'X' is Gaussian distributed

    if X<=norminv(Gamma,mu,sigma) % H0 True
        %fprintf('x = %.2f <= %.2f --> Epsilon is white --> The model is valid\n', X,norminv(G

    else % H1 True
        %fprintf('x = %.2f > %.2f --> Epsilon is NOT white --> The model is NOT valid\n', X,no
        H1 = H1+1;
    end
end
```

To make the final decision we can use the *Anderson test,* where $\bar{m}$ (in the code H_1) is the number of failed tests

$$\begin{cases} H_0 : \dfrac{\bar{m}}{m} \leq \alpha & \varepsilon\left(t, \widehat{\theta_N}\right) \text{ is zero mean white process} \\ H_1 : \dfrac{\bar{m}}{m} > \alpha & \text{not } H_0 \end{cases}$$

```
if (H1/(m-1))<=Alpha % H0
```

```
    fprintf('%.3f <= %.2f --> Handerson test passed --> The model is valid\n', (H1/(m-1)),Alpha
else % H1
    fprintf('%.3f > %.2f --> Handerson test NOT passed --> The model is NOT valid\n', (H1/(m-1)
end
```

```
0.011 <= 0.05 --> Handerson test passed --> The model is valid
```

```
0; % just a breakpoint
```

Exercise 2: Classification

LEDS Exam: September 11, 2023

Then, the classification problem should be solved following the steps below:

1. **Understand the feature set in space**: The provided training features are scattered in space in a particular (geometrical) way. Understand it and use it to define a "refined" feature set if needed. (Plotting them may result useful). Notice that the training set distribution in space changes at every run of the function, however the boundary shape is always the same. This can be handled easily in a well thought program.
2. **Define the required classification algorithm:** describe the algorithm you plan to use and why and add brief description of the matlab function you implemented accordingly.
3. **Verify and test your classificator:** verify the obtained classificator on the provided training set and compute the related classification error. Then, try your classificator on the test set and plot your results together with the classification error on the test set.

```
clear all;
close all;
addpath ./Functions;
```

1] System measurements

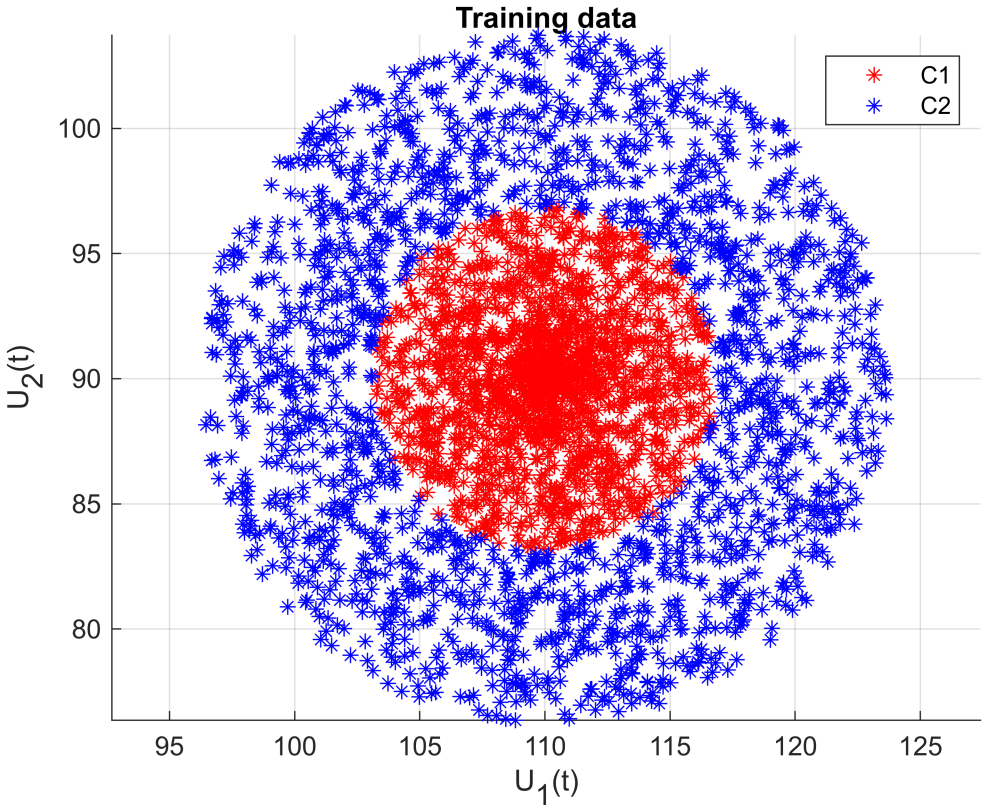
```
N=5000;
Student = 'Simone Cenerini';
Matriculation = '0000983677';
[FeatureSet] = ClassifyThose(N,Student,Matriculation);
```

```
Utrain  = FeatureSet.Utrain;
Ytrain  = FeatureSet.Ytrain;
Utest   = FeatureSet.Utest;
Ytest   = FeatureSet.Ytest;
```

2] Data analysis:

First of all we plot the obtained data in order to understand the geometrical shape of the data.

```
N_train = size(Utrain, 1);
dimension = size(Utrain, 2);
N_test = size(Utest, 1);

Plot_classification_data(Ytrain,Utrain,'U_1(t)','U_2(t)');
```



As we can easily see from the plot we are dealing with a *non linear problem* (the input space is non linear) and in particular the shape of the decision boundary is a circle.

2.1] Non linear trasformation of the input space:

The decision boundary is a circle with a centre of coordinates

$$\text{Centre} = \left(C_{U_1}; C_{U_2}\right)$$

What we want to do is to linearize the input space and so redefine a new linear input space in order to have a linear decision boundary easier to be found.

With the following piece of code we try fo find the centre of the circular decision boundary by doing the aritmetic mean of the values of the input points.

```
Centre(1) = mean(Utrain(:,1));
Centre(2) = mean(Utrain(:,2));
fprintf('The centre is C = (%.2f,%.2f)\n', Centre(1), Centre(2));
```

```
The centre is C = (110.03,89.98)
```

Now we can introduce our non linear trasformation of the input space in order to obtain a linear boundary.

$u(t)\,\epsilon\,U \longrightarrow v(t) = g(u(t))\,\epsilon\,V$

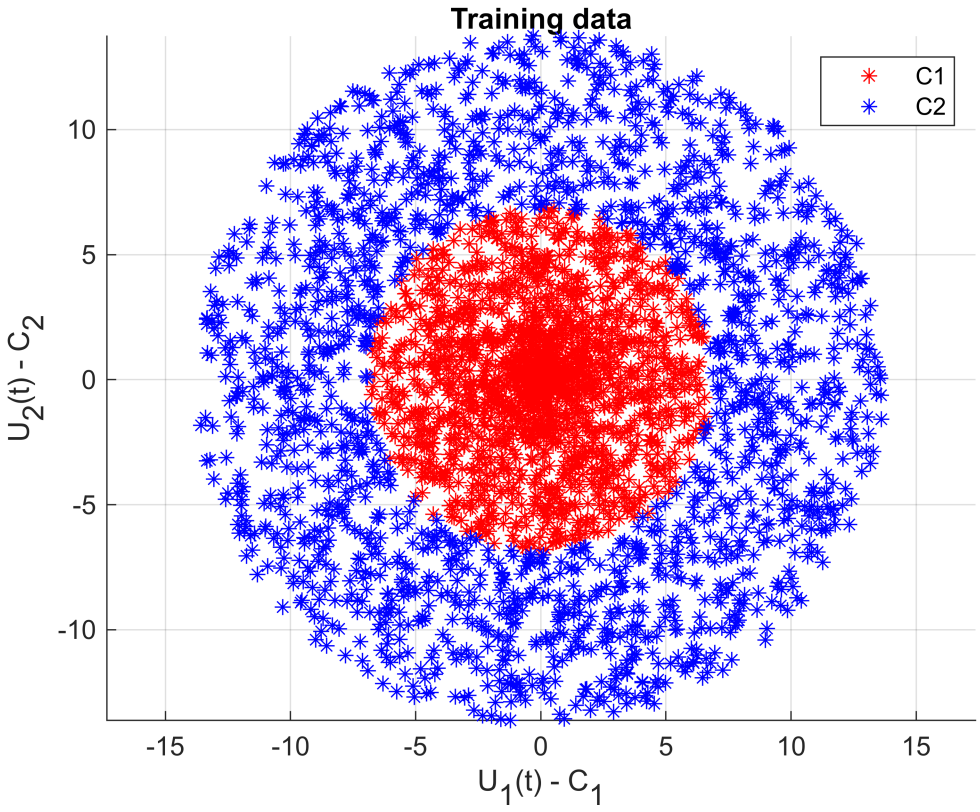
with $g(\cdot)$ non linear transformation.

$$u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \longrightarrow v(t) = \begin{bmatrix} (u_1 - C_{U_1})^2 \\ (u_2 - C_{U_2})^2 \end{bmatrix}$$

in our particular case both the "original" and the linearized input spaces have the same dimension. Usually the dimension increase.

Firstly we centrilize the training set:

```
U_centered_train(:,1) = Utrain(:,1)-Centre(1);
U_centered_train(:,2) = Utrain(:,2)-Centre(2);

Plot_classification_data(Ytrain,U_centered_train,'U_1(t) - C_1','U_2(t) - C_2');
```



Since during the project we encountered an issue with the cost function (which will be explained later on), one attempt was also made by normalizing the input samples to observe potential benefits on the sigmoid function and Hessian of the loss function.

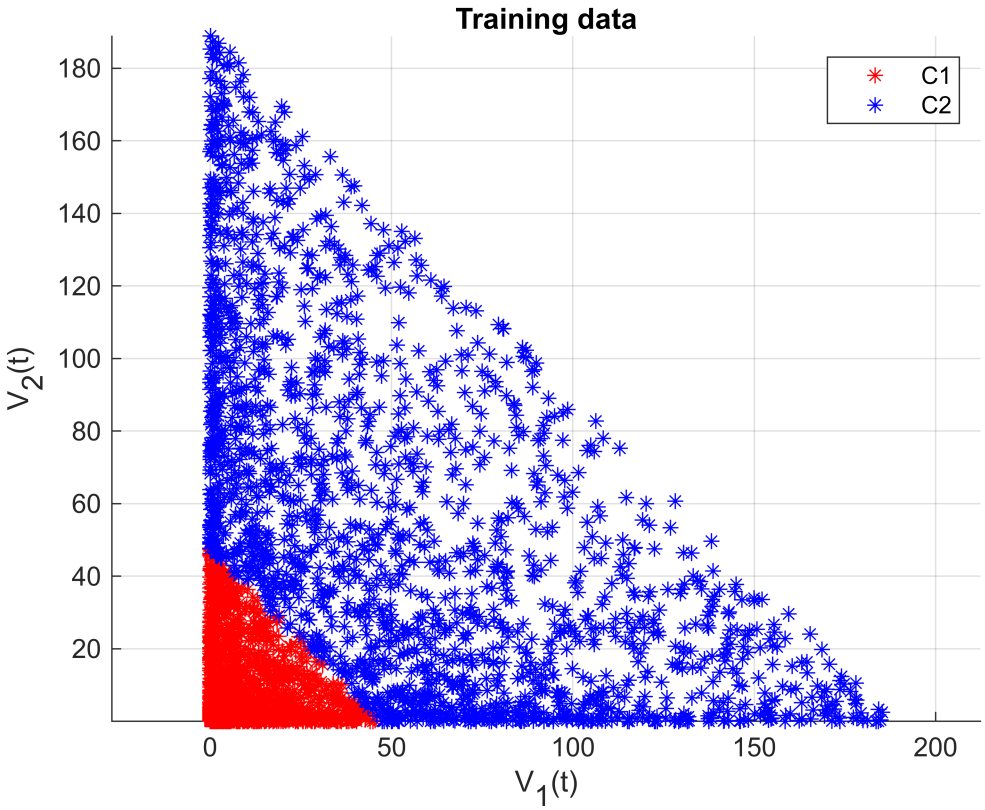
However in conclusion of the project, a better result on error rate is achieved without samples normalization, but we need to be carefull with the choise of the initial condition and the stepsize.

To use normalized data change the value of *Normalize_Data* to *{ true, false}.*

```
Normalize_Data = false;

if Normalize_Data == false
    U_scalated_train(:,1) = U_centered_train(:,1);
    U_scalated_train(:,2) = U_centered_train(:,2);
elseif Normalize_Data == true
    U_scalated_train(:,1) = U_centered_train(:,1)/(max(U_centered_train(:,1)));
    U_scalated_train(:,2) = U_centered_train(:,2)/(max(U_centered_train(:,2)));
    Plot_classification_data(Ytrain,U_scalated_train,'U_1_s(t)','U_2_s(t)');
end
```

In conclusion we effectivly introduce the non linear transformation:

```
Vtrain(:,1) = U_scalated_train(:,1).*U_scalated_train(:,1);
Vtrain(:,2) = U_scalated_train(:,2).*U_scalated_train(:,2);

Plot_classification_data(Ytrain,Vtrain,'V_1(t)','V_2(t)');
```



3] Define the required classification algorithm:

Now, thanks to our transformation, we are dealing with a linear classfication problem, and we are asked to solve this problem by means of **logistic regression**.

So our objective are the posterior probabilities $P(C_1|u(t))$ and $P(C_2|u(t)) = 1 - P(C_1|u(t))$.

These probabilities are found using the following logistic sigmoid function

$$f(z) = \frac{e^z}{1+e^z},$$

where $z(t) = \varphi^T(t)\theta$ with $\varphi(t) = [1 \;\; v_1(t) \;\; v_2(t)]^T$ and $\theta = [\beta_0 \;\; \beta_1 \;\; \beta_2]^T$

In fact our decision boundary has the following form: $\beta_0 + \beta_1 \; v_1(t) + \beta_2 \; v_2(t) = 0$.

Our estimate of the parameters $\widehat{\theta}$ can be found by maximizing the following maximum likelihood function

$$P(Y|\theta) = \prod_{t=1}^{N} P(C_1|v(t))^{y(t)} \, (1 - P(C_1|v(t)))^{1-y(t)} = \prod_{t=1}^{N} f(z(t))^{y(t)} \, f(z(t))^{1-y(t)}$$

that can be written in logaritmic form as

$$log \, P(Y|\theta) = \sum_{t=1}^{N} [y(t) \, log \, f(z(t)) + (1 - y(t)) \, log \, (1 - f(z(t)))]$$

Maximize the above function is equivalent to minimize its negative, so we have to find the estimate $\widehat{\theta}$ minimizing $J(\theta) = -log \, P(Y|\theta)$

The algorithm we use to find our estimate $\widehat{\theta}$ is the recursive **Newton-Raphson algorithm**:

$$\widehat{\theta}^{k+1} = \widehat{\theta}^k - \eta^k J''\left(\widehat{\theta}^k\right)^{-1} J'\left(\widehat{\theta}^k\right)^T$$

where $J'(\theta) = \sum_{t=1}^{N} [f(z(t)) - y(t)] \, \varphi(t) = \; \Phi^T(F(\theta) - Y)$

and $J''(\theta) = \sum_{t=1}^{N} f(z(t))(1 - f(z(t))) \, \varphi(t) \, \varphi^T(t) = \; \Phi^T W(\theta)\Phi$

where $W(\theta) = \begin{bmatrix} f(z(1))(1-f(z(1))) & 0 & \cdots & 0 \\ 0 & f(z(2))(1-f(z(2))) & \cdots & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & f(z(N))(1-f(z(N))) \end{bmatrix}$

The stopping criterion fo the algorithm is the following

$$\left(\frac{\|\hat{\theta}^{k+1}-\hat{\theta}^{k}\|}{\|\hat{\theta}^{k}\|}<\delta\right) \parallel (k>\bar{k})$$

We implemented the Newton-Raphson method starting with an initial Theta represented by a zero vector. The step size was chosen to decrease gradually in order to achieve faster results.

As mentioned earlier, during the project, we encountered an issue with the cost function. After improving the accuracy of Theta, the cost function resulted in 'Not a Number' (NaN). This occurred due to the logarithmic function when $f(z)$ give an informatical 1 or 0.

We manage this problem by introduce $\varepsilon \approx 0$ inseide the logaritmic funcion, to ensure it never resulted in NaN.

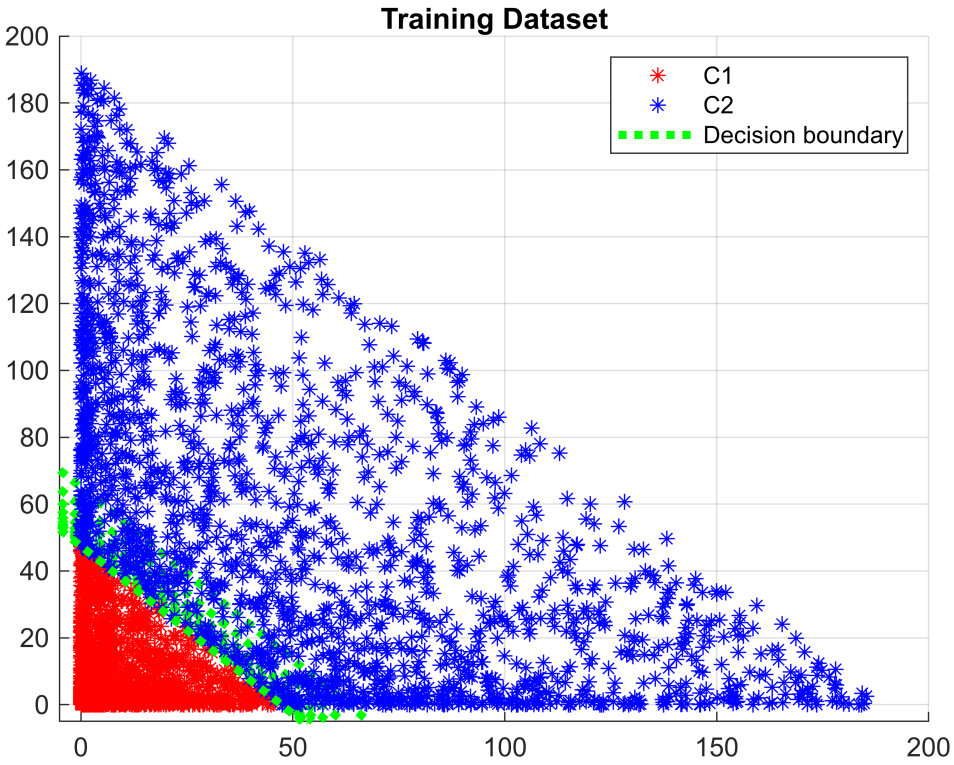
```
% Parameter & Costant
Plotting = true;
MAX_ITER = 200; % k bar
Theta_0 = [0;0;0];
Stopping_value = 0.00001;

% Algorithm
Phi = [ones(N_train,1) Vtrain(:,1) Vtrain(:,2)];
Y(find(Ytrain==2),1) = 1;
Y(find(Ytrain==1),1) = 0;
Theta = zeros(MAX_ITER,dimension+1);
Theta(1,:) = Theta_0;
J = zeros(MAX_ITER,1);
J_dt = zeros(MAX_ITER,dimension+1);

Plot_class = figure();
Plot_CostFun = figure();
Stopping_algorithm = false;
for k = 1:MAX_ITER
    stepsize = (1/(k^0.5)); %(1/k^a) with a = [0.5,1[
    %stepsize = 0.1;

    J(k) = LRCostFunc(Y,Phi,Theta(k,:)');
    J_dt(k,:) = LRCostFuncGrad(Y,Phi,Theta(k,:)');
    J_ddt = LRCostFuncHessian(Y,Phi,Theta(k,:)');
    Descent = (inv(J_ddt)*(J_dt(k,:)'))';

    Theta(k+1,:) = Theta(k,:) - stepsize*Descent;

    Stopping_criteria = norm(Theta(k+1)-Theta(k))/norm(Theta(k));

    Norm_J_dt(k) = norm(J_dt(k,:));
    fprintf('k=%d\tStpz= %0.3f\tJ= %.2f\t|J_dt|= %.4f\t|J_ddt|= %.2f\t|D|= %.2f\tUpdateTheta= %
        ,k,stepsize,J(k),Norm_J_dt(k),norm(J_ddt),norm(Descent),Stopping_criteria);

    if Stopping_criteria < Stopping_value
```

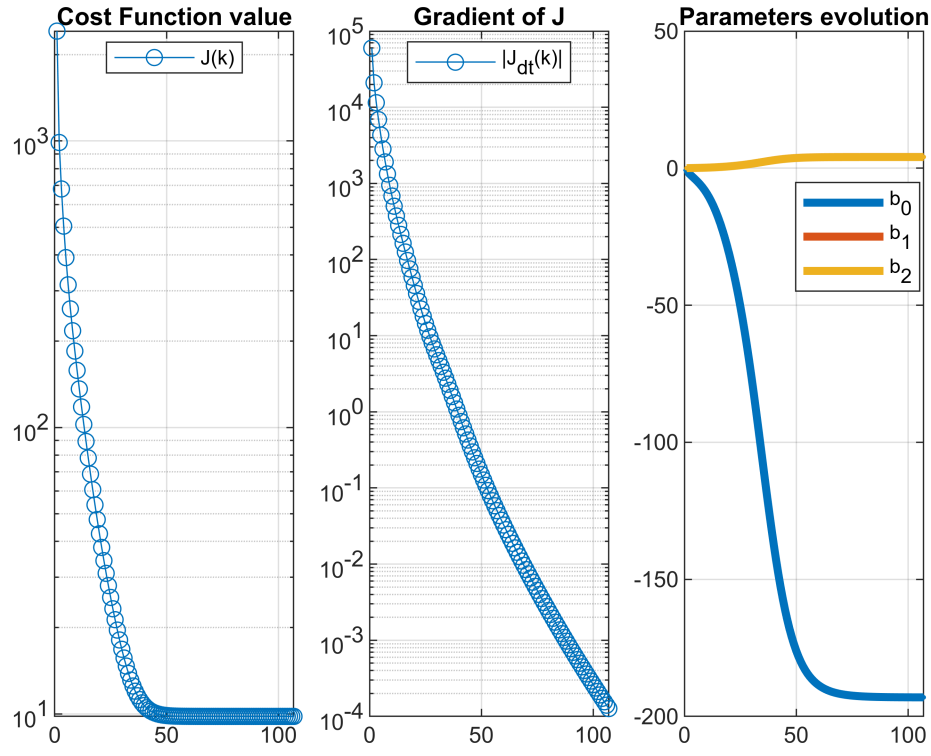
```
        Stopping_algorithm = true;
    end

    if (Plotting == true) && (mod(k,10)==0 || k<10 || Stopping_algorithm == true)
        %subplot(2,3,[1 3]);
        figure(Plot_class);
        hold on
        grid on
        plot(Vtrain(find(Ytrain==1),1),Vtrain(find(Ytrain==1),2),'r*'); %C1
        plot(Vtrain(find(Ytrain==2),1),Vtrain(find(Ytrain==2),2),'b*'); %C2
        Xlim=xlim();
        xspace = Xlim(1):0.1:Xlim(end);
        BoundaryLine = (Theta(k,1) + xspace.*Theta(k,2))./(-Theta(k,3));
        plot(xspace,BoundaryLine,':g','LineWidth',3);
        Ylim=ylim();
        Ylim(1)=-5;
        ylim(Ylim);
        Xlim(1)=-5;
        xlim(Xlim);
        if Normalize_Data == true
            xlim([-0.1,1.1]);
            ylim([-0.1,1.1]);
        end
        hold off
        title('Training Dataset');
        legend({'C1','C2','Decision boundary'});

        figure(Plot_CostFun);
        subplot(1,3,1);
        semilogy(1:k,J(1:k),'o-');
        grid on;
        title('Cost Function value');
        legend({'J(k)'},'Location', 'Best');

        subplot(1,3,2);
        semilogy(1:k,Norm_J_dt(1:k),'o-');
        grid on;
        title('Gradient of J');
        legend({'|J_d_t(k)|'},'Location', 'Best');

        subplot(1,3,3);
        plot(1:k,Theta(1:k,:),'.-','LineWidth',3);
        grid on;
        title('Parameters evolution');
        legend({'b_0','b_1','b_2'},'Location', 'Best');

        pause(0.01);
    end

    if Stopping_algorithm == true;
```

```
        break
    end
end
```

```
k=1    Stpz= 1.000    J= 2419.02    |J_dt|= 59840.4097    |J_ddt|= 3261097.21    |D|= 1.88    UpdateTheta= Inf
k=2    Stpz= 0.707    J= 988.06    |J_dt|= 20960.3222    |J_ddt|= 1355053.36    |D|= 1.13    UpdateTheta= 0.426731
k=3    Stpz= 0.577    J= 680.05    |J_dt|= 11482.5093    |J_ddt|= 717936.76    |D|= 1.29    UpdateTheta= 0.279000
k=4    Stpz= 0.500    J= 504.42    |J_dt|= 6870.3747    |J_ddt|= 449895.83    |D|= 1.54    UpdateTheta= 0.224638
k=5    Stpz= 0.447    J= 391.65    |J_dt|= 4328.4168    |J_ddt|= 315524.96    |D|= 1.83    UpdateTheta= 0.195240
k=6    Stpz= 0.408    J= 314.70    |J_dt|= 2834.2546    |J_ddt|= 238607.33    |D|= 2.18    UpdateTheta= 0.177418
k=7    Stpz= 0.378    J= 259.34    |J_dt|= 1915.3079    |J_ddt|= 189724.01    |D|= 2.59    UpdateTheta= 0.166218
k=8    Stpz= 0.354    J= 217.55    |J_dt|= 1329.0949    |J_ddt|= 155925.04    |D|= 3.09    UpdateTheta= 0.158936
k=9    Stpz= 0.333    J= 184.73    |J_dt|= 943.2946    |J_ddt|= 130953.93    |D|= 3.68    UpdateTheta= 0.153773
k=10    Stpz= 0.316    J= 158.16    |J_dt|= 682.3523    |J_ddt|= 111578.05    |D|= 4.35    UpdateTheta= 0.149504
k=11    Stpz= 0.302    J= 136.23    |J_dt|= 501.5351    |J_ddt|= 96027.59    |D|= 5.10    UpdateTheta= 0.145408
k=12    Stpz= 0.289    J= 117.89    |J_dt|= 373.5371    |J_ddt|= 83266.85    |D|= 5.92    UpdateTheta= 0.141201
k=13    Stpz= 0.277    J= 102.44    |J_dt|= 281.2293    |J_ddt|= 72637.73    |D|= 6.82    UpdateTheta= 0.136855
k=14    Stpz= 0.267    J= 89.35    |J_dt|= 213.5900    |J_ddt|= 63687.09    |D|= 7.79    UpdateTheta= 0.132426
k=15    Stpz= 0.258    J= 78.22    |J_dt|= 163.3554    |J_ddt|= 56085.17    |D|= 8.82    UpdateTheta= 0.127967
k=16    Stpz= 0.250    J= 68.73    |J_dt|= 125.6312    |J_ddt|= 49584.88    |D|= 9.92    UpdateTheta= 0.123517
k=17    Stpz= 0.243    J= 60.61    |J_dt|= 97.0506    |J_ddt|= 43997.83    |D|= 11.08    UpdateTheta= 0.119116
k=18    Stpz= 0.236    J= 53.65    |J_dt|= 75.2528    |J_ddt|= 39177.59    |D|= 12.30    UpdateTheta= 0.114820
k=19    Stpz= 0.229    J= 47.68    |J_dt|= 58.5517    |J_ddt|= 35007.21    |D|= 13.58    UpdateTheta= 0.110697
k=20    Stpz= 0.224    J= 42.54    |J_dt|= 45.7210    |J_ddt|= 31390.39    |D|= 14.93    UpdateTheta= 0.106817
k=21    Stpz= 0.218    J= 38.11    |J_dt|= 35.8523    |J_ddt|= 28245.61    |D|= 16.36    UpdateTheta= 0.103227
k=22    Stpz= 0.213    J= 34.27    |J_dt|= 28.2614    |J_ddt|= 25502.72    |D|= 17.89    UpdateTheta= 0.099945
k=23    Stpz= 0.209    J= 30.94    |J_dt|= 22.4239    |J_ddt|= 23101.23    |D|= 19.52    UpdateTheta= 0.096951
k=24    Stpz= 0.204    J= 28.04    |J_dt|= 17.9331    |J_ddt|= 20989.40    |D|= 21.25    UpdateTheta= 0.094191
k=25    Stpz= 0.200    J= 25.50    |J_dt|= 14.4712    |J_ddt|= 19123.73    |D|= 23.07    UpdateTheta= 0.091587
k=26    Stpz= 0.196    J= 23.27    |J_dt|= 11.7889    |J_ddt|= 17468.31    |D|= 24.97    UpdateTheta= 0.089046
k=27    Stpz= 0.192    J= 21.32    |J_dt|= 9.6927    |J_ddt|= 15994.02    |D|= 26.91    UpdateTheta= 0.086473
k=28    Stpz= 0.189    J= 19.61    |J_dt|= 8.0340    |J_ddt|= 14677.47    |D|= 28.85    UpdateTheta= 0.083784
k=29    Stpz= 0.186    J= 18.11    |J_dt|= 6.7011    |J_ddt|= 13499.95    |D|= 30.73    UpdateTheta= 0.080914
k=30    Stpz= 0.183    J= 16.80    |J_dt|= 5.6121    |J_ddt|= 12446.24    |D|= 32.49    UpdateTheta= 0.077821
k=31    Stpz= 0.180    J= 15.67    |J_dt|= 4.7089    |J_ddt|= 11503.77    |D|= 34.08    UpdateTheta= 0.074490
k=32    Stpz= 0.177    J= 14.68    |J_dt|= 3.9507    |J_ddt|= 10661.76    |D|= 35.42    UpdateTheta= 0.070931
k=33    Stpz= 0.174    J= 13.84    |J_dt|= 3.3094    |J_ddt|= 9910.78    |D|= 36.48    UpdateTheta= 0.067173
k=34    Stpz= 0.171    J= 13.12    |J_dt|= 2.7655    |J_ddt|= 9242.33    |D|= 37.21    UpdateTheta= 0.063255
k=35    Stpz= 0.169    J= 12.51    |J_dt|= 2.3045    |J_ddt|= 8648.63    |D|= 37.59    UpdateTheta= 0.059227
k=36    Stpz= 0.167    J= 12.00    |J_dt|= 1.9152    |J_ddt|= 8122.54    |D|= 37.59    UpdateTheta= 0.055139
k=37    Stpz= 0.164    J= 11.57    |J_dt|= 1.5880    |J_ddt|= 7657.44    |D|= 37.22    UpdateTheta= 0.051040
k=38    Stpz= 0.162    J= 11.22    |J_dt|= 1.3147    |J_ddt|= 7247.20    |D|= 36.49    UpdateTheta= 0.046979
k=39    Stpz= 0.160    J= 10.93    |J_dt|= 1.0875    |J_ddt|= 6886.18    |D|= 35.43    UpdateTheta= 0.042998
k=40    Stpz= 0.158    J= 10.69    |J_dt|= 0.8994    |J_ddt|= 6569.18    |D|= 34.06    UpdateTheta= 0.039139
k=41    Stpz= 0.156    J= 10.50    |J_dt|= 0.7444    |J_ddt|= 6291.43    |D|= 32.44    UpdateTheta= 0.035434
k=42    Stpz= 0.154    J= 10.35    |J_dt|= 0.6168    |J_ddt|= 6048.58    |D|= 30.62    UpdateTheta= 0.031914
k=43    Stpz= 0.152    J= 10.23    |J_dt|= 0.5120    |J_ddt|= 5836.66    |D|= 28.66    UpdateTheta= 0.028602
k=44    Stpz= 0.151    J= 10.13    |J_dt|= 0.4260    |J_ddt|= 5652.08    |D|= 26.60    UpdateTheta= 0.025516
k=45    Stpz= 0.149    J= 10.05    |J_dt|= 0.3553    |J_ddt|= 5491.59    |D|= 24.50    UpdateTheta= 0.022665
k=46    Stpz= 0.147    J= 9.99    |J_dt|= 0.2972    |J_ddt|= 5352.26    |D|= 22.42    UpdateTheta= 0.020054
k=47    Stpz= 0.146    J= 9.95    |J_dt|= 0.2493    |J_ddt|= 5231.46    |D|= 20.38    UpdateTheta= 0.017682
k=48    Stpz= 0.144    J= 9.91    |J_dt|= 0.2098    |J_ddt|= 5126.86    |D|= 18.42    UpdateTheta= 0.015542
k=49    Stpz= 0.143    J= 9.89    |J_dt|= 0.1770    |J_ddt|= 5036.36    |D|= 16.57    UpdateTheta= 0.013625
k=50    Stpz= 0.141    J= 9.87    |J_dt|= 0.1499    |J_ddt|= 4958.13    |D|= 14.84    UpdateTheta= 0.011917
k=51    Stpz= 0.140    J= 9.85    |J_dt|= 0.1272    |J_ddt|= 4890.54    |D|= 13.24    UpdateTheta= 0.010403
k=52    Stpz= 0.139    J= 9.84    |J_dt|= 0.1083    |J_ddt|= 4832.17    |D|= 11.78    UpdateTheta= 0.009068
k=53    Stpz= 0.137    J= 9.83    |J_dt|= 0.0925    |J_ddt|= 4781.76    |D|= 10.44    UpdateTheta= 0.007895
k=54    Stpz= 0.136    J= 9.82    |J_dt|= 0.0792    |J_ddt|= 4738.23    |D|= 9.24    UpdateTheta= 0.006867
k=55    Stpz= 0.135    J= 9.82    |J_dt|= 0.0679    |J_ddt|= 4700.64    |D|= 8.16    UpdateTheta= 0.005969
k=56    Stpz= 0.134    J= 9.81    |J_dt|= 0.0584    |J_ddt|= 4668.17    |D|= 7.20    UpdateTheta= 0.005187
k=57    Stpz= 0.132    J= 9.81    |J_dt|= 0.0503    |J_ddt|= 4640.10    |D|= 6.34    UpdateTheta= 0.004506
k=58    Stpz= 0.131    J= 9.81    |J_dt|= 0.0435    |J_ddt|= 4615.84    |D|= 5.58    UpdateTheta= 0.003915
k=59    Stpz= 0.130    J= 9.80    |J_dt|= 0.0376    |J_ddt|= 4594.84    |D|= 4.91    UpdateTheta= 0.003401
k=60    Stpz= 0.129    J= 9.80    |J_dt|= 0.0326    |J_ddt|= 4576.66    |D|= 4.32    UpdateTheta= 0.002956
```

```
k=61    Stpz= 0.128    J= 9.80    |J_dt|= 0.0283    |J_ddt|= 4560.91    |D|= 3.80    UpdateTheta= 0.002570
k=62    Stpz= 0.127    J= 9.80    |J_dt|= 0.0246    |J_ddt|= 4547.26    |D|= 3.34    UpdateTheta= 0.002236
k=63    Stpz= 0.126    J= 9.80    |J_dt|= 0.0215    |J_ddt|= 4535.40    |D|= 2.94    UpdateTheta= 0.001946
k=64    Stpz= 0.125    J= 9.80    |J_dt|= 0.0187    |J_ddt|= 4525.11    |D|= 2.58    UpdateTheta= 0.001695
k=65    Stpz= 0.124    J= 9.80    |J_dt|= 0.0163    |J_ddt|= 4516.16    |D|= 2.27    UpdateTheta= 0.001477
k=66    Stpz= 0.123    J= 9.80    |J_dt|= 0.0143    |J_ddt|= 4508.37    |D|= 2.00    UpdateTheta= 0.001288
k=67    Stpz= 0.122    J= 9.80    |J_dt|= 0.0125    |J_ddt|= 4501.59    |D|= 1.76    UpdateTheta= 0.001124
k=68    Stpz= 0.121    J= 9.80    |J_dt|= 0.0110    |J_ddt|= 4495.67    |D|= 1.55    UpdateTheta= 0.000982
k=69    Stpz= 0.120    J= 9.80    |J_dt|= 0.0096    |J_ddt|= 4490.51    |D|= 1.37    UpdateTheta= 0.000859
k=70    Stpz= 0.120    J= 9.80    |J_dt|= 0.0085    |J_ddt|= 4486.01    |D|= 1.21    UpdateTheta= 0.000751
k=71    Stpz= 0.119    J= 9.80    |J_dt|= 0.0075    |J_ddt|= 4482.07    |D|= 1.07    UpdateTheta= 0.000658
k=72    Stpz= 0.118    J= 9.80    |J_dt|= 0.0066    |J_ddt|= 4478.62    |D|= 0.94    UpdateTheta= 0.000577
k=73    Stpz= 0.117    J= 9.80    |J_dt|= 0.0058    |J_ddt|= 4475.60    |D|= 0.83    UpdateTheta= 0.000506
k=74    Stpz= 0.116    J= 9.80    |J_dt|= 0.0051    |J_ddt|= 4472.95    |D|= 0.74    UpdateTheta= 0.000444
k=75    Stpz= 0.115    J= 9.80    |J_dt|= 0.0045    |J_ddt|= 4470.62    |D|= 0.65    UpdateTheta= 0.000390
k=76    Stpz= 0.115    J= 9.80    |J_dt|= 0.0040    |J_ddt|= 4468.58    |D|= 0.58    UpdateTheta= 0.000343
k=77    Stpz= 0.114    J= 9.80    |J_dt|= 0.0035    |J_ddt|= 4466.79    |D|= 0.51    UpdateTheta= 0.000302
k=78    Stpz= 0.113    J= 9.80    |J_dt|= 0.0031    |J_ddt|= 4465.21    |D|= 0.45    UpdateTheta= 0.000266
k=79    Stpz= 0.113    J= 9.80    |J_dt|= 0.0028    |J_ddt|= 4463.82    |D|= 0.40    UpdateTheta= 0.000235
k=80    Stpz= 0.112    J= 9.80    |J_dt|= 0.0025    |J_ddt|= 4462.59    |D|= 0.36    UpdateTheta= 0.000207
k=81    Stpz= 0.111    J= 9.80    |J_dt|= 0.0022    |J_ddt|= 4461.51    |D|= 0.32    UpdateTheta= 0.000183
k=82    Stpz= 0.110    J= 9.80    |J_dt|= 0.0019    |J_ddt|= 4460.55    |D|= 0.28    UpdateTheta= 0.000162
k=83    Stpz= 0.110    J= 9.80    |J_dt|= 0.0017    |J_ddt|= 4459.71    |D|= 0.25    UpdateTheta= 0.000143
k=84    Stpz= 0.109    J= 9.80    |J_dt|= 0.0015    |J_ddt|= 4458.96    |D|= 0.22    UpdateTheta= 0.000127
k=85    Stpz= 0.108    J= 9.80    |J_dt|= 0.0014    |J_ddt|= 4458.30    |D|= 0.20    UpdateTheta= 0.000112
k=86    Stpz= 0.108    J= 9.80    |J_dt|= 0.0012    |J_ddt|= 4457.72    |D|= 0.18    UpdateTheta= 0.000099
k=87    Stpz= 0.107    J= 9.80    |J_dt|= 0.0011    |J_ddt|= 4457.20    |D|= 0.16    UpdateTheta= 0.000088
k=88    Stpz= 0.107    J= 9.80    |J_dt|= 0.0010    |J_ddt|= 4456.74    |D|= 0.14    UpdateTheta= 0.000078
k=89    Stpz= 0.106    J= 9.80    |J_dt|= 0.0009    |J_ddt|= 4456.33    |D|= 0.13    UpdateTheta= 0.000070
k=90    Stpz= 0.105    J= 9.80    |J_dt|= 0.0008    |J_ddt|= 4455.97    |D|= 0.11    UpdateTheta= 0.000062
k=91    Stpz= 0.105    J= 9.80    |J_dt|= 0.0007    |J_ddt|= 4455.65    |D|= 0.10    UpdateTheta= 0.000055
k=92    Stpz= 0.104    J= 9.80    |J_dt|= 0.0006    |J_ddt|= 4455.36    |D|= 0.09    UpdateTheta= 0.000049
k=93    Stpz= 0.104    J= 9.80    |J_dt|= 0.0006    |J_ddt|= 4455.10    |D|= 0.08    UpdateTheta= 0.000044
k=94    Stpz= 0.103    J= 9.80    |J_dt|= 0.0005    |J_ddt|= 4454.88    |D|= 0.07    UpdateTheta= 0.000039
k=95    Stpz= 0.103    J= 9.80    |J_dt|= 0.0004    |J_ddt|= 4454.67    |D|= 0.07    UpdateTheta= 0.000035
k=96    Stpz= 0.102    J= 9.80    |J_dt|= 0.0004    |J_ddt|= 4454.49    |D|= 0.06    UpdateTheta= 0.000031
k=97    Stpz= 0.102    J= 9.80    |J_dt|= 0.0004    |J_ddt|= 4454.33    |D|= 0.05    UpdateTheta= 0.000028
k=98    Stpz= 0.101    J= 9.80    |J_dt|= 0.0003    |J_ddt|= 4454.19    |D|= 0.05    UpdateTheta= 0.000025
k=99    Stpz= 0.101    J= 9.80    |J_dt|= 0.0003    |J_ddt|= 4454.06    |D|= 0.04    UpdateTheta= 0.000022
k=100    Stpz= 0.100    J= 9.80    |J_dt|= 0.0003    |J_ddt|= 4453.94    |D|= 0.04    UpdateTheta= 0.000020
k=101    Stpz= 0.100    J= 9.80    |J_dt|= 0.0002    |J_ddt|= 4453.84    |D|= 0.03    UpdateTheta= 0.000018
k=102    Stpz= 0.099    J= 9.80    |J_dt|= 0.0002    |J_ddt|= 4453.74    |D|= 0.03    UpdateTheta= 0.000016
k=103    Stpz= 0.099    J= 9.80    |J_dt|= 0.0002    |J_ddt|= 4453.66    |D|= 0.03    UpdateTheta= 0.000014
k=104    Stpz= 0.098    J= 9.80    |J_dt|= 0.0002    |J_ddt|= 4453.59    |D|= 0.03    UpdateTheta= 0.000013
k=105    Stpz= 0.098    J= 9.80    |J_dt|= 0.0002    |J_ddt|= 4453.52    |D|= 0.02    UpdateTheta= 0.000011
k=106    Stpz= 0.097    J= 9.80    |J_dt|= 0.0001    |J_ddt|= 4453.46    |D|= 0.02    UpdateTheta= 0.000010
k=107    Stpz= 0.097    J= 9.80    |J_dt|= 0.0001    |J_ddt|= 4453.41    |D|= 0.02    UpdateTheta= 0.000009
```





We obtained a great result !!

The cost function reaches its minimum around 10. The gradient of the cost function approaches zero, indicating that we have perfectly achieved the minimum, and it decreases in a quadratic manner. Additionally, the descent direction almost reaches zero, indicating that we are precisely at the minimum.

To make the optimal Theta values more understandable, we need to scale them by dividing them by their maximum value.

```
Theta_star = Theta(k+1,:)';
Theta_star = Theta_star/max(Theta_star);
fprintf('The estimated parameters are [%.2f %.2f %.2f] \nWitch define the decision boundary %.2
    Theta_star(1),Theta_star(2),Theta_star(3),Theta_star(1),Theta_star(2),Theta_star(3));
```

```
The estimated parameters are [-47.28 1.00 1.00]
Witch define the decision boundary -47.28 + 1.00 V_1(t) + 1.00 V_2(t) = 0
```

4] Verify and test your classificator:

In order to test the performances of our classifier we take into account the error rate that has, in the more general case, the following definition:

$$E_R = \frac{1}{N}\sum_{t=1}^{N} I(y(t), \hat{y}(t)) \text{ where } I(y(t), \hat{y}(t)) = \begin{cases} 1 & IF\ y(t) \neq \hat{y}(t) \\ 0 & IF\ y(t) = \hat{y}(t) \end{cases}$$

so, it counts the number of missclassified observations.

So the **classification error** over the *training set* is:

```
F = sigmoid(Phi*Theta_star);
Y_hat_train = round(F);

Er_train = (1/N_train) * sum(abs(Y-Y_hat_train));
fprintf('Error Rate on the training set = %2.2f %%',Er_train*100);
```

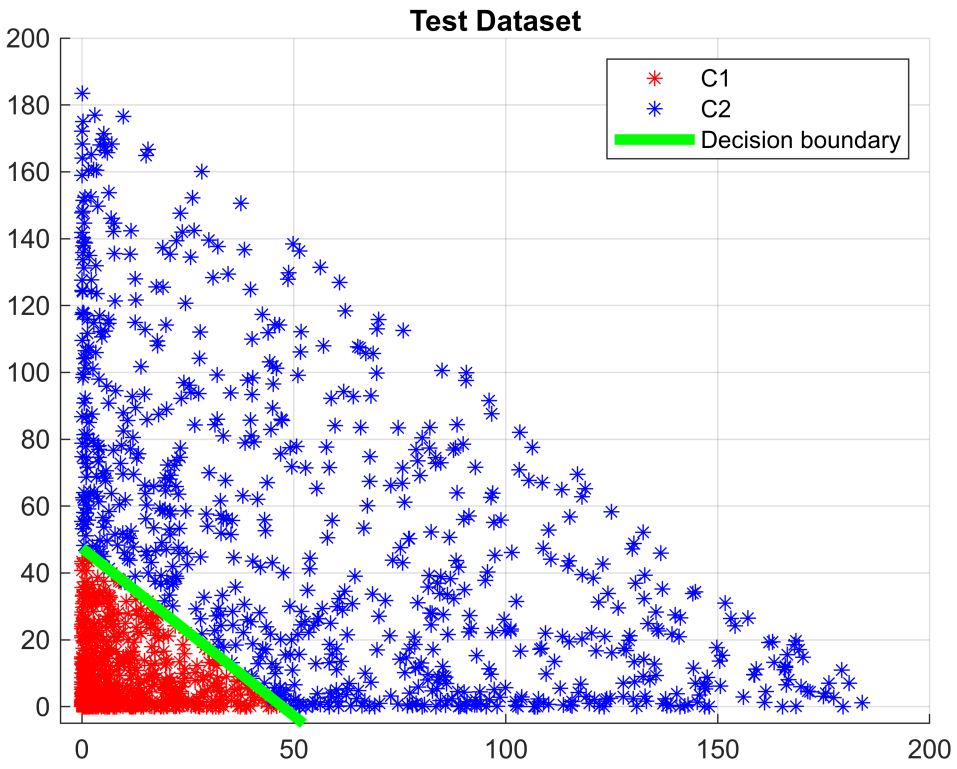
```
Error Rate on the training set = 0.11 %
```

4.1] Classification on the Test set:

Again, the test set also needs to be nonlinearly transformed, but using the same function applied to the training set. For this reason, we should use the center obtained from the training set and it is maximun value.

```
if Normalize_Data == false
    Vtest(:,1) = ((Utest(:,1)-Centre(1))) .* ((Utest(:,1)-Centre(1)));
    Vtest(:,2) = ((Utest(:,2)-Centre(2))) .* ((Utest(:,2)-Centre(2)));
elseif Normalize_Data == true
    Centre_test(1) = Centre(1);
    Centre_test(2) = Centre(2);
    Vtest(:,1) = ((Utest(:,1)-Centre_test(1))./(max(U_centered_train(:,1)))) .* ((Utest(:,1)-Ce
    Vtest(:,2) = ((Utest(:,2)-Centre_test(2))./(max(U_centered_train(:,1)))) .* ((Utest(:,2)-Ce
end

figure();
```

```
hold on
grid on
plot(Vtest(find(Ytest==1),1),Vtest(find(Ytest==1),2),'r*'); %C1
plot(Vtest(find(Ytest==2),1),Vtest(find(Ytest==2),2),'b*'); %C2
Xlim=xlim();
xspace = Xlim(1):0.1:Xlim(end);
BoundaryLine = (Theta_star(1) + xspace.*Theta_star(2))./(-Theta_star(3));
plot(xspace,BoundaryLine,'-g','LineWidth',4);
Ylim=ylim();
Ylim(1)=-5;
ylim(Ylim);
Xlim(1)=-5;
xlim(Xlim);
if Normalize_Data == true
    xlim([-0.1,1.1]);
    ylim([-0.1,1.1]);
end
hold off
title('Test Dataset');
legend({'C1','C2','Decision boundary'});
```



So the **classification error** over the *training set* is:

```
Phi_test = [ones(N_test,1) Vtest(:,1) Vtest(:,2)];
Y_test(find(Ytest==2),1) = 1;
Y_test(find(Ytest==1),1) = 0;
```

```
F_test = sigmoid(Phi_test*Theta_star);
Y_hat_test = round(F_test);

Er_test = (1/N_test) * sum(abs(Y_test-Y_hat_test));
fprintf('Error Rate on the test set = %2.2f %%',Er_test*100);
```

```
Error Rate on the test set = 0.20 %
```

5] MATLAB mnrfit function:

As we had done in the "ML_Ex1" matlab homework I wanto to compare the result obtained with the MATLAB mnrfit function.

```
Theta_matlab = -mnrfit(Vtrain,Ytrain);
Theta_matlab = Theta_matlab/max(Theta_matlab);
Phi = [ones(N_train,1) Vtrain(:,1) Vtrain(:,2)];

F_matlab = sigmoid(Phi*Theta_matlab);
Y_hat_train_matlab = round(F_matlab);

Er_train_matlab = (1/N_train) * sum(abs(Y-Y_hat_train_matlab));
fprintf(['Error Rate on the training set with MATLAB function = %2.10f %%\n' ...
    'Error Rate on the training set with above algorithm = %2.10f %%'],Er_train_matlab*100, Er_
```

```
Error Rate on the training set with MATLAB function = 0.1142857143 %
Error Rate on the training set with above algorithm = 0.1142857143 %
```

That is equal to the one we have obtained before.

The code performes like MatLab! :)