```
clear
clc
close all
```

**OTTIMIZZAZIONE**

```
%Definisco il numero di variabili decisionali
n= ;
%Definisco la matrice A dei coefficienti dei vincoli di disuguaglianza
A=[];
%Definisco la matrice b dei termini noti dei vincoli di disuguaglianza
b=[];
%Definisco la matrice A_eq dei coefficienti dei vincoli di uguaglianza
A_eq=[];
%Definisco la matrice b_eq dei termini noti dei vincoli di uguaglianza
b_eq=[];
%se non ci sono vincoli di uguaglianza le matrici A_eq e b_eq posso o non
scriverle o lasciarle vuote
%Definisco LB (lower bound) e UB (upper bound)
LB=[]; %se nel testo non è specificato metto -Inf tante volte quante sono
le variabili decisionali
UB=[]; %se nel testo non è specificato metto Inf tante volte quante sono
le variabili decisionali
%Definisco la condizione iniziale randomica
X0=rand(n,1);
[X,fval,exitflag]=fmincon(@(x)my_obj(x),X0,A,b,A_eq,b_eq,LB,UB,@(x)my_nl(
x))
disp('The optimal solution is:')
disp(X);
%in un altro script
function f=my_obj(x)
f= %copio la funzione obiettivo
end
%in un altro script
function [c c_eq]=my_nl(x)
c= ; %copio il vincolo non lineare passando il termine noto al primo
membro (attenzione lo copio in c se il vincolo non lineare è un vincolo
di disuguaglianza, se non c'è un vincolo di disuguaglianza non lineare
scrivo c=[];
c_eq= ; %copio il vincolo non lineare passando il termine noto al primo
membro (attenzione lo copio in c_eq se il vincolo non lineare è un
vincolo di uguaglianza, se non c'è un vincolo di uguaglianza non lineare
scrivo c_eq=[];
end
```

**REGRESSIONE 2D**

```
load data_2d.mat

n=200;         %totale dati (te lo da il workspace)
N=150;         %dati su cui costruisco il modello (a scelta basta che N<n)

%plottiamo i dati
figure
scatter(X,y,'filled');
hold on

%divido il dataset in training set e test set
X_train=X(1:N,:);
X_test=X(N+1:end,:);
y_train=y(1:N,:);
y_test=y(N+1:end,:);

%REGRESSIONE AI MINIMI QUADRATI
%Costruiamo PHI dei regressori
PHI=[X_train ones(N,1)];

%faccio la stima
theta_ls=PHI\y_train;

%faccio la previsione
y_hat=PHI*theta_ls;

%plottiamo la retta di regressione
plot(X_train,y_hat,'r','LineWidth',2)
xlabel('Feature')
ylabel('Target')
hold off;

%statistiche sul training set
y_hat=[X_train ones(N,1)] * theta_ls;  %y predette
err_train=y_hat-y_train(:);
TSS_train=sum((y_train-mean(y_train)).^2); %total sum of squares
RSS_train=sum(err_train.^2); %residual sum of squares
Rsq_train = 1-RSS_train/TSS_train; %R-squared
MSE_train = RSS_train/N % Mean Squared Error

% statistiche sul test set
y_pred = [X_test ones(n-N,1)] * theta_ls; % y predette
err_test = y_pred-y_test(:);
TSS_test = sum((y_test-mean(y_test)).^2);
RSS_test = sum(err_test.^2);
Rsq_test = 1-RSS_test/TSS_test;
MSE_test = RSS_test/N                  %oppure diviso n??
MSE_test1= min(err_test.^2)            %altro modo per MSE
```

**REGRESSIONE 3D**

```
load data_3d.mat

n = 200; % numero totale di dati (te lo da il workspace)
N = 150; %dati su cui costruisco il modello (a scelta basta che N<n)

% plottiamo i dati
figure;
scatter3(X(:,1),X(:,2), y, 'filled'); % scatter3 sviluppa il grafico in 3 dimensioni
hold on;

% dividiamo il dataset in training e test set
X_train = X(1:N,:);
X_test = X(N+1:end,:);
y_train = y(1:N,:);
y_test = y(N+1:end,:);

% costruisco la matrice PHI dei regressori
PHI = [X_train ones(N,1)];

% faccio la stima
theta_ls = PHI\y_train;

% plottiamo il piano di regressione
a = linspace(min(X(:,1)), max(X(:,1)), 50)'; %vettore di valori sull'asse x
b = linspace(min(X(:,2)), max(X(:,2)), 50)'; %vettore di vaalori sull'asse y
[xx,yy] = meshgrid(a, b); % si crea una griglia
Z = reshape([xx(:), yy(:), ones(size(xx(:)))] * theta_ls, numel(a), []);
surf(xx, yy, Z, 'FaceAlpha', 0.5, 'EdgeColor','none')
xlabel('Feature 1')
ylabel('Feature 2')
zlabel('Target')
hold off;

% statistiche sul training set
y_hat = [X_train(:,1) X_train(:,2) ones(N,1)] * theta_ls; % y predette
err_train = y_hat-y_train(:); % errore: vettore y - y cappello
TSS_train = sum((y_train-mean(y_train)).^2); % varianza campionaria di y
% oppure TSS=(y_train-mean(y_train))'*(y_train-mean(y_train))
RSS_train = sum(err_train.^2); % norma dell'errore di previsione
% oppure RSS_train = err_train'*err_train
Rsq_train = 1-RSS_train/TSS_train

% statistiche sul test set
y_pred = [X_test(:,1) X_test(:,2) ones(n-N,1)] * theta_ls; % y predette
err_test = y_pred-y_test(:);
TSS_test = sum((y_test-mean(y_test)).^2);
RSS_test = sum(err_test.^2);
Rsq_test = 1-RSS_test/TSS_test

% essendo Rsq_train e Rsq_test molto vicini il nostro modello è molto utile.
```

**IDENTIFICAZIONE ARMA**

```
%% IDENTIFICAZIONE DI UN ARMA
load my_id_b.mat

%grafichiamo la serie storica
figure(1)
plot(y_ARMA)

%autocorrelazione totale che va a 0 o infinito
figure(2)
autocorr(y_ARMA)

%autocorrelazione parziale che va a 0 o infinito
figure(3)
parcorr(y_ARMA)

%ARMA totale e parziale non ho pallini su zero
%AR totale non ho pallini su zero e parziale si
%MA totale pallini sullo zero e parziale no

%nel workspace vedo che y_ARMA è 5100x1
N=3000; %dati che utilizziamo (N<5100)

%inizializzo il vettore y tenendo conto del possibile transitorio
y=y_ARMA(101:N+100);

%identifico la componente AR con metodo di Yule-Walker --> stimo le
%autocorrelazioni
[acf,lags,bounds] = autocorr(y);

%non conosco l'ordine quindi procedo per tentativi
n_D=3;

%inizializzo la matrice n_Dxn_D come matrice di zeri
R=zeros(n_D);

%scartiamo ro di 0 dal vettore acf
acf=acf(2:end);

% costruiamo la matrice R per colonna:
for kk = 1:n_D
    R(:,kk) = acf(n_D+1-kk:2*n_D-kk); %ogni riga deve corrispondere a eq di Yule Walker
end

% il termine noto
my_rho = acf(n_D + 1:2*n_D);

% facciamo la stima risolvendo il sistema di equazioni
theta_AR = R\my_rho;

% facciamo la stima risolvendo il sistema di equazioni
theta_AR = R\my_rho;

%controllo se l'ordine ipotizzato è corretto
for kk = 1:n_D
    PHI_AR(:,kk) = y(n_D+1-kk:N-kk);
end

y_hat_AR = PHI_AR*theta_AR;
```

```
% la stima della componente MA è quindi
my_eta = y(n_D+1:N)-y_hat_AR;

% grafichimo i correlogrammi di my_eta e mi fermo coi tentativi su n_D quando
% trovo correlogrammi del MA (totale va su zero e parziale no)
figure(4)
autocorr(my_eta)
figure(5)
parcorr(my_eta)

%% identificazione della componente MA di un ARMA (*)
% l'ordine tentativo:
n_eq = 17;

%parto da n=x del pallino in figure5 che precede il primo pallino nell'intervallo di
confidenza

% un AR si identifica ai minimi quadrati quindi dobbiamo definire PHI. (uso
% for perché vado per tentativi)

for kk = 1:n_eq
    PHI_eq(:,kk) = my_eta(n_eq+1-kk:end-kk);
end

% i coefficiente l'AR equivalente:
theta_eq = PHI_eq\my_eta(n_eq+1:end);

% la previsione dell'uscita di un AR, che in questo caso è eta:
eta_hat = PHI_eq*theta_eq;

w_hat = my_eta(n_eq+1:end)-eta_hat;        %parte di rumore bianco del modello
% per vedere se w_hat ha le caratteristiche statistiche di un rumore bianco
% grafico la funzione di autocorrelazione totale di w_hat e faccio il test
% di bianchezza:

figure(6)
autocorr(w_hat)
[h_eta, p_eta] = lbqtest(w_hat);   % all'esame fermati quae poi riparti da my_eta
quando trovi il giusto n_eq
p=p_eta
if p>0.05
    disp('ok')
else
    disp('No')
end
% cambio l'ordine di tentativo fino a quando le autocorrelazioni sforano e
% fino a quando non passo il test di bianchezza (cioè quando p>0.05).
% a n_eq= 16 passo il test ma per poco quindi aumento ancora e con n_eq=17
% siamo tranquilli.

my_eta = my_eta(n_eq+1:end); % così dimentico di non avere le w corrispondenti
% l'ordine del polinomio lo conosciamo perchè eta è la parte MA, i suoi
% correlogrammi mi danno l'ordine del polinomio N, infatti guardando la
% figura 4 vedo che l'ordine è 3.

n_N = 3;    %X della stanghetta prima di entrare nell'intervallo di confidenza in
figure4

% dobbiamo fare dinuovo la stima ai minimi quadrati, quindi costruiamo la
% matrice PHI, i regressori sono w_hat
```

```
for kk = 1:n_N
   PHI_MA(:,kk) = w_hat(n_N+1-kk:end-kk);
end

% il termine noto è eta-w, la stima dei coefficienti MA:
theta_MA = PHI_MA\(my_eta(n_N+1:end)-w_hat(n_N+1:end))

% (se dovessimo fare l'identificazione solo della componente MA il codice
% inizierebbe da qui (*) ).
```

**IDENTIFICAZIONE AR**

```
load data_AR.mat         %carico i dati y_AR

figure(1)
plot(y_AR)          %grafico della serie storica
title('Grafico della serie storica')

figure(2)
autocorr(y_AR)      %autocorrelazione totale che va a 0 o Infinito

figure(3)
parcorr(y_AR)       %autocorrelazione parziale

% mi accorgo che è un AR(3)
y=y_AR(101:end);          %butto via i primi 100 (a caso) dati perché potrebbero
rappresentare il transitorio

N=1500; %dati che utilizziamo
n_D=3; %ordine del modello

%sto risolvendo con minimi quadrati quindi costruisco matrice PHI
PHI = [y(n_D:N-1),y(n_D-1:N-2),y(n_D-2:N-3)];
%scrivo il vettore dei parametri stimati
theta_hat=PHI\y(n_D+1:N);
my_eps=y(n_D+1:N)-PHI*theta_hat

figure(4)
autocorr(my_eps)   %autocorrelazione totale del rumore

%test di bianchezza
[H,P] = lbqtest(my_eps)  %se P<0.95 non riggetto la statistica, viceversa la rigetto

if P<0.95
    disp('Non rigetto la statistica')
else
    disp('Rigetto la statistica')
end

varianza_y=std(y)
varianza_errore=std(my_eps)

if std(my_eps)<std(y)
    disp('Il modello mi da vantaggi in termini di previsione')
else
    disp('Il modello non mi da vantaggi in termini di previsione')
end
```

SIMULAZIONE LTI/RUMORE BIANCO

```
%dimensione dello stato e matrici
n=2;

A=[.4 -.2;.1 .6];
B=[1 0]';
C=[1 1];

%definisco il tempo di simulazione
T=50;

%preallochiamo il vettore di stato come una matrice di
zeri da riempire
x=zeros(n,T);

%CI
x(:,1)=[10 -5]';

for kk=1:T-1
    x(:,kk+1)=A*x(:,kk)+B*randn(1);
    y(kk)=C*x(:,kk);
end

figure
plot(x','-*','LineWidth',2)
eig(A)
```

**MARKOVITZ CLASSICO COME A LEZIONE**

```
% definisco il numero di asset
n = 4;
% definisco il numero di variabili dipendenti
m = 2;

% definisco la matrice di covarianza degli investimenti
S_r = [.5   -0.1  0     -0.05;
       -0.1 .4    -0.05 0;
       0    -0.05  0.6   0;
       -0.05   0    0     0.1];
% definisco il vettore riga dei rendimenti attesi
m_r = [0.08 0.04 0.06 0.02];
% definisco il vettore uno trasposto
uno_t = ones(1,n);

% definisco la matrice dei vincoli del problema
A = [uno_t; m_r];

% definisco i blocchi della matrice dei vincoli relativi a variabili
% dipendenti (A_x) e variabili indipendenti (A_w)
A_x = A(:,1:m);
A_w = A(:,m+1:n);

% definisco il vettore dei termini noti
b = [1; 0.012];

% Definisco i blocchi della matrice S_r
S_xx = S_r(1:m,1:m);
S_ww = S_r(m+1:n,m+1:n);

S_xw = S_r(1:m,m+1:n);
S_wx = S_xw';

% Implemento la soluzione analitica per trovare il calcolo delle variabili indipendenti
w_star = (A_w'*(A_x')^(-1)*S_xx*(A_x)^(-1)*A_w-S_wx*(A_x)^(-1)*A_w...
    -A_w'*(A_x')^(-1)*S_xw+S_ww)/(A_w'*(A_x')^(-1)*S_xx*(A_x)^(-1)...
    -S_wx*(A_x)^(-1))*b;

% calcolo le variabili dipendenti
x_star = -(A_x)^(-1)*A_w*w_star+ (A_x)^(-1)*b;

% calcolo gli autovalori dell'Hessiana del problema non vincolato ottenuto
% sostituendo i vincoli nella funzione obiettivo.
eig(A_w'*(A_x')^(-1)*S_xx*(A_x)^(-1)*A_w-S_wx*(A_x)^(-1)*A_w...
    -A_w'*(A_x')^(-1)*S_xw+S_ww)

%% calcolo della frontiera di efficienza
% per calcolare la frontiera di efficienza risolvo il problema per diversi
% valori del rendimento atteso del portafogli
my_rho = [0.000002:.000002:0.02];

for kk = 1:length(my_rho)
    % per ogni valore del rendimento atteso in my_rho
    b = [1; my_rho(kk)];
    % Implemento la soluzione analitica per trovare il calcolo delle variabili
indipendenti
```

```matlab
    w_star = (A_w'*(A_x')^(-1)*S_xx*(A_x)^(-1)*A_w-S_wx*(A_x)^(-1)*A_w...
        -A_w'*(A_x')^(-1)*S_xw+S_ww)/(A_w'*(A_x')^(-1)*S_xx*(A_x)^(-1)...
        -S_wx*(A_x)^(-1))*b

    % calcolo le variabili dipendenti
    x_star = -(A_x)^(-1)*A_w*w_star+ (A_x)^(-1)*b
    % calcolo la varianza del portafogli sostituendo la soluzione ottima
    % nella cifra di merito
    my_risk(kk) = [x_star' w_star']*S_r*[x_star; w_star];

end

plot(my_rho,my_risk,'LineWidth',2)
title('Frontiera di Efficienza','FontSize',32)
xlabel('$\rho$','Interpreter','latex','FontSize',24)
ylabel('$\sigma^2_w$','Interpreter','latex','FontSize',24)
```

**OTTIMIZZAZIONE CON MARKOVITZ**

**Esercizio 1. Ottimizzazione del portafoglio alla Markovitz**

Dato un capitale a disposizione pari a 1, risolvere il problema di ottimizzazione che permetta di costruire un portafoglio a minima varianza costituito da i titoli Enel, Ferrari e Intesa San Paolo imponendo un rendimento atteso desiderato pari a 0.1.

Si conoscono inoltre i rendimenti attesi dei tre titoli pari rispettivamente a $m_r = [0.3, 0.5, 0.4]$ e la seguente matrice di covarianza

$$\Sigma = \begin{bmatrix} 0.10 & 0.02 & -0.04 \\ 0.02 & 0.12 & 0 \\ -0.04 & 0 & 0.11 \end{bmatrix}$$

```
%definisco il numero di variabili ed i dati
n=3;
rendimenti_attesi=[.3 .5 .4];
covarianza=[.10 .02 -0.04;.02 .12 0;-0.4 0 .11];
b=[];     %rendimento atteso minimo richiesto per il portafoglio

LB=zeros(n,1);
X0=zeros(n,1);

%vincoli
A=[];

Aeq=[rendimenti_attesi;ones(1,n)];
beq=[0.1;1];

options=optimoptions('fmincon','Algorithm','interior-point','Display','iter-
detailed','PlotFcn','optimplotx')

%ottimizzazione markovitz
[X,fval,exitflag]=fmincon(@(x)x'*covarianza*x,X0,A,b,Aeq,beq,LB,[],[],options)
```

# Esercizio ottimizzazione vincolata

April 3, 2023

Let us consider a manufacturing firm that produces 5 different types of smartphones using the following resources: labor, materials and energy.
The objective of the firm is to find an optimal trade-off between the amount of product and the total cost of production subject to the following constraints:

- The total labor burden cannot exceed 500 hours per week
- The total materials input cannot exceed 10,000 units per week
- The total energy cost cannot exceed 5000 units per week
- The total production capacity of the firm cannot exceed 7000 units per week
- The firm must produce at least 1000 units of each type of smartphone

The trade-off between the amount of product and its cost is measured by a quadratic function of the amount $u_1, u_2, u_3, u_4, u_5$ of the five items produced by the firm. Considering the unit coefficients associated to the per unit resource requirements of the products leads to formulating the following optimization problem

$$\underset{u}{\text{minimize}} \quad L(u) = -\mathbf{1}^T\mathbf{u} + 0.5u_1^2 + 0.7u_2^2 + 0.8u_3^2 + 1.2u_4^2 + 1.5u_5^2 \tag{1a}$$

subject to

$$0.1u_1 + 0.12u_2 + 0.08u_3 + 0.07u_4 + 0.06u_5 \leq 500, \tag{1b}$$
$$1.7u_1 + 0.9u_2 + 1.8u_3 + 1.6u_4 + 1.4u_5 \leq 10000, \tag{1c}$$
$$0.5u_1 + 0.8u_2 + 0.6u_3 + 0.7u_4 + u_5 \leq 5000, \tag{1d}$$
$$u_1 + u_2 + u_3 + u_4 + u_5 \leq 7000, \tag{1e}$$
$$u_i \geq 1000 \qquad \forall i = 1, \ldots, 5 \tag{1f}$$

```
%definisco numero variabili decisionali
n=5;   %u senza pedice è tutto il vettore

%definisco matrici dei vincoli di diseg lineari
A=[.1 .12 .08 .07 .06; 1.7 .9 1.8 1.6 1.4; .5 .8 .6 .7 1; 1 1 1 1 1];
b=[500 10000 5000 7000]';

%vincoli di uguaglianza lineari non ce ne sono
A_eq=[];
b_eq=[];

%LowerBound e UpperBound  (con ones perché vale per tutte le variabili)
LB=1000*ones(n,1);
UB=7000*ones(n,1);          %perché se la somma deve fare 7000 al massimo ognuna può fare
7000

%scelgo come condizione iniziale il LB per obbligare matlab a usare il
%metodo di barriera
u_0=1000*ones(n,1);      %potevo anche fare =rand(n,1)

%il seguente vettore può non mettersi ma consente di settare un certo
%algoritmo per la soluzione del problema vedendo tutte le iterazioni

options=optimoptions('fmincon','Algorithm','interior-point','Display','Iter-
detailed','PlotFcn','optimplotfval');

%risolviamo il problema di ottimizzazione

[u,fval,exitflag]=fmincon(@(u)funzione_obiettivo(u),u_0,A,b,A_eq,b_eq,LB,UB,[],options)
;

%stampiamo la soluzione
disp('The optimal solution is:')
disp(u)
disp(['The minimum cost of production is $' num2str(fval)])



function L=funzione_obiettivo(u)
L=-u'*u+0.5*u(1)^2+0.7*u(2)^2+0.8*u(3)^2+1.2*u(4)^2+1.5*u(5)^2;
end
```

**PROPAGAZIONE RUMORE BIANCO**

**Esercizio 1. Propagazione del rumore**

Simulare l'evoluzione di un sistema lineare tempo invariante a tempo discreto del tipo

$$x(k+1) = Ax(k) + Bv(k),$$
$$y(k) = Cx(k)$$

dove $v(k)$ è un rumore bianco. Scegliere

- la matrice $A$ in maniera tale che il sistema abbia 3 variabili di stato, e che il suo stato sia un processo asintoticamente stazionario;
- e le matrici $B$ e $C$ in maniera tale che il sistema abbia un unico ingresso e una unica uscita.

```
%variabili di stato
n=3;

%scelgo matrice A in modo che il sistema sia asintoticamente stabile
A=[.1 0 0;0 .2 0;0 0 .3];
%matrici B e C ok sempre così
B=[1 0 0]';
C=[1 1 1];
%verifico che autovalori siano in modulo minori di 1
eig(A)

%tempo di simulazione
T=50;

%prealloco x come matrice di zeri
x=zeros(n,T);

%CI
x(:,1)=10*rand(3,1);

for kk=1:T-1
    x(:,kk+1)=A*x(:,kk)+B*randn(1);
    y(kk)=C*x(:,kk);
end

figure(1)
plot(x','-*','LineWidth',2)
title('Stato')
xlabel('k')
ylabel('x(k)')
grid on

figure(2)
plot(y,'-*','LineWidth',2)
title('Uscita')
xlabel('k')
ylabel('y(k)')
grid on
```