

# Controllo della temperatura di una casa

## Indice

Descrizione del problema.....	1
Parametri costanti del modello della casa.....	3
Equazioni del sistema in variabili di stato.....	3
Vincoli su stato e ingresso.....	5
Punti di equilibrio.....	5
Sistema linearizzato.....	5
Autovalori del sistema linearizzato a tempo continuo.....	7
Matrici del sistema linearizzato a tempo discreto.....	8
Espressione dei vincoli nelle coordinate del sistema linearizzato.....	9
Calcolo del Control Invariant Set.....	9
Formulazione del problema quadratico.....	10
Formulazione del controllore MPC.....	13
Analisi delle prestazioni del controllore MPC progettato.....	13
Funzioni.....	14

## Descrizione del problema

L'obiettivo del progetto è regolare la temperatura di una casa utilizzando un controllore MPC. Si ipotizza che la casa non condivida alcun muro esterno con altre abitazioni, che i flussi d'aria tra le stanze della casa siano trascurabili e che il riscaldamento della casa sia affidato a tre termosifoni, uno per ogni stanza. La pianta della casa in oggetto è mostrata nella figura seguente:

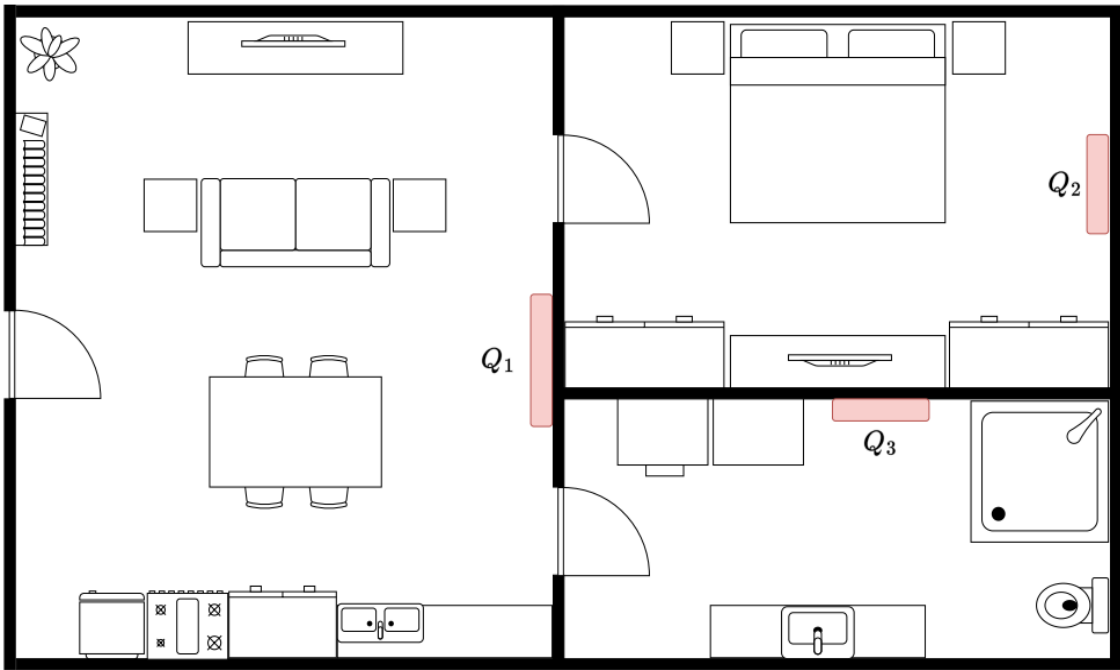


Figura 1: Planimetria della casa.

La dinamica della temperatura all'interno della casa è definita dalle equazioni seguenti:

$$\left\{ \begin{array}{l} C_1 \dot{T}_1(t) = Q_1(t) - k_{1,2}(t)(T_1(t) - T_2(t)) - k_{1,3}(t)(T_1(t) - T_3(t)) - k_{ext}(T_1(t) - T_{ext}) \\ C_2 \dot{T}_2(t) = Q_2(t) + k_{1,2}(t)(T_1(t) - T_2(t)) - k_{2,3}(t)(T_2(t) - T_3(t)) - k_{ext}(T_2(t) - T_{ext}) \\ C_3 \dot{T}_3(t) = Q_3(t) + k_{1,3}(t)(T_1(t) - T_3(t)) + k_{2,3}(t)(T_2(t) - T_3(t)) - k_{ext}(T_3(t) - T_{ext}) \end{array} \right. \quad \text{con } k_{i,j} = \bar{k}_{i,j} + \frac{4}{1 + e^{-0.5\|T_i(t) - T_j(t)\|_2}}$$

$$\begin{array}{l} \dot{Q}_1 = (Q_{1,r}(t) - Q_1(t))/\tau_1 \\ \dot{Q}_2 = (Q_{2,r}(t) - Q_2(t))/\tau_2 \\ \dot{Q}_3 = (Q_{3,r}(t) - Q_3(t))/\tau_3 \end{array}$$

Il significato delle grandezze presenti nelle precedenti equazioni è il seguente:

- $T_i [K]$  è la temperatura della i-esima stanza;
- $T_{ext} [K]$  è la temperatura esterna, ipotizzata costante.
- $C_i [J/s]$  è la capacità termica della i-esima stanza;
- $k_{i,j} [W/K]$  è un parametro che regola lo scambio di calore tra l'i-esima e la j-esima stanza;
- $k_{ext} [W/K]$  è un parametro che regola lo scambio di calore tra le stanze della casa e l'esterno;
- $Q_i [W]$  è la potenza termica dell'i-esimo termosifone;
- $Q_{i,r} [W]$  è la potenza termica di riferimento dell'i-esimo termosifone;
- $\tau_i [s]$  è la costante di tempo dell'i-esimo termosifone.

I parametri noti sono riportati nella tabella seguente:

Parametro	Valore	Unità
$C_1$	6800	$J/s$
$C_2$	5600	$J/s$
$C_3$	5100	$J/s$
$\bar{k}_{1,2}$	15	$W/K$
$\bar{k}_{1,3}$	20	$W/K$
$\bar{k}_{2,3}$	15	$W/K$
$k_{ext}$	8	$W/K$
$T_{ext}$	278	$K$
$\tau_1$	400	$s$
$\tau_2$	450	$s$
$\tau_3$	450	$s$

Tabella 1: Parametri noti del sistema in esame.

La potenza erogata dai tre termosifoni è non-negativa e mai superiore ai 150W, mentre la temperatura all'interno delle tre stanze non deve mai scendere al di sotto dei 286K.

Gli ingressi controllati del sistema sono le potenze di riferimento dei tre termosifoni,  $Q_{i,r}$ ,  $i \in \{1, 2, 3\}$ .

L'obiettivo del lavoro è progettare un controllore MPC in grado di portare il sistema dalla condizione iniziale  $(T_1, T_2, T_3, Q_1, Q_2, Q_3) = (288, 288, 288, 0, 0, 0)$  all'equilibrio  $(293, 293, 293, 120, 120, 120)$ , rispettando sempre i vincoli, e simulare il funzionamento del sistema in anello chiuso. Il tempo minimo di campionamento,  $T_s$ , è pari ad 1 s.

## Parametri costanti del modello della casa

Inizialmente, si creano delle variabili per salvare tutti i parametri noti del modello.

```
clc
close all
clear
T_ext = 278; % [K], temperatura esterna ipotizzata costante
C = [6800 5600 5100]; % [J/s], capacità termica della i-esima stanza
k_bar = [15 20 15]; % [W/K], scambio di calore fra l'i-esima e la j-esima stanza
k_ext = 8; % [W/K], scambio di calore fra le stanze e l'esterno
Q_ref = [150 150 150]; % [W], potenza termica di riferimento dell'i-esimo termosifone
tau = [400 450 450]; % [s], costante di tempo dell'i-esimo termosifone
x_eq = [293 293 293 120 120 120]';
x_ci = [288 288 288 0 0 0]';
Ts = 1; % [s], tempo di campionamento minimo
```

## Equazioni del sistema in variabili di stato

Prima di procedere ad analizzare il sistema, è necessario riscrivere le equazioni della sua dinamica nello spazio degli stati. Si definiscono il vettore degli stati,  $\mathbf{x}$ , ed il vettore degli ingressi,  $\mathbf{u}$ :

$$\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6]' := [T_1, T_2, T_3, Q_1, Q_2, Q_3]', \quad \mathbf{x} \in \mathbb{R}^6$$

$$\mathbf{u} = [u_1, u_2, u_3]' := [Q_{1,r}, Q_{2,r}, Q_{3,r}]', \quad \mathbf{u} \in \mathbb{R}^3$$

Il sistema diventa quindi:

$$\begin{cases} \dot{x}_1(t) = (x_4(t) - k_{1,2}(t)(x_1(t) - x_2(t)) - k_{1,3}(t)(x_1(t) - x_3(t)) - k_{ext}(x_1(t) - T_{ext}))/C_1 \\ \dot{x}_2(t) = (x_5(t) + k_{1,2}(t)(x_1(t) - x_2(t)) - k_{2,3}(t)(x_2(t) - x_3(t)) - k_{ext}(x_2(t) - T_{ext}))/C_2 \\ \dot{x}_3(t) = (x_6(t) + k_{1,3}(t)(x_1(t) - x_3(t)) + k_{2,3}(t)(x_2(t) - x_3(t)) - k_{ext}(x_3(t) - T_{ext}))/C_3 \\ \dot{x}_4 = (u_1(t) - x_4(t))/\tau_1 \\ \dot{x}_5 = (u_2(t) - x_5(t))/\tau_2 \\ \dot{x}_6 = (u_3(t) - x_6(t))/\tau_3 \end{cases} \quad \text{con } k_{i,j} = \bar{k}_{i,j} + \frac{4}{1 + e^{-0.5\|x_i(t) - x_j(t)\|_2}}$$

Le equazioni vengono scritte in Matlab utilizzando la notazione simbolica.

```
syms x1 x2 x3 x4 x5 x6 x1d x2d x3d x4d x5d x6d u1 u2 u3;
% Coefficienti k(i,j)
% Definizione dei vettori per poter calcolare correttamente le norme
x1_vect = [x1;0;0];
x2_vect = [0;x2;0];
x3_vect = [0;0;x3];
```

```

k12 = k_bar(1)+(4/(1+exp(-0.5*(norm(x1_vect-x2_vect,2))^2)));
k13 = k_bar(2)+(4/(1+exp(-0.5*(norm(x1_vect-x3_vect,2))^2)));
k23 = k_bar(3)+(4/(1+exp(-0.5*(norm(x2_vect-x3_vect,2))^2)));
% Equazioni della dinamica del sistema
eq1 = x1d == (x4 -k12*(x1-x2) -k13*(x1-x3) -k_ext*(x1-T_ext))/C(1);
eq2 = x2d == (x5 +k12*(x1-x2) -k23*(x2-x3) -k_ext*(x2-T_ext))/C(2);
eq3 = x3d == (x6 +k13*(x1-x3) +k23*(x2-x3) -k_ext*(x3-T_ext))/C(3);
eq4 = x4d == (u1-x4)/tau(1);
eq5 = x5d == (u2-x5)/tau(2);
eq6 = x6d == (u3-x6)/tau(3);
equazioni = [eq1;eq2;eq3;eq4;eq5;eq6]

```

equazioni =

$$\begin{pmatrix} x1d = \frac{x_4}{6800} - \frac{x_1}{850} - \frac{\sigma_3}{6800} - \frac{\sigma_1}{6800} + \frac{139}{425} \\ x2d = \frac{x_5}{5600} - \frac{x_2}{700} + \frac{\sigma_3}{5600} - \frac{\sigma_2}{5600} + \frac{139}{350} \\ x3d = \frac{x_6}{5100} - \frac{2x_3}{1275} + \frac{\sigma_2}{5100} + \frac{\sigma_1}{5100} + \frac{556}{1275} \\ x4d = \frac{u_1}{400} - \frac{x_4}{400} \\ x5d = \frac{u_2}{450} - \frac{x_5}{450} \\ x6d = \frac{u_3}{450} - \frac{x_6}{450} \end{pmatrix}$$

where

$$\sigma_1 = \left( \frac{4}{e^{-\sigma_6 - \sigma_4} + 1} + 20 \right) (x_1 - x_3)$$

$$\sigma_2 = \left( \frac{4}{e^{-\sigma_5 - \sigma_4} + 1} + 15 \right) (x_2 - x_3)$$

$$\sigma_3 = \left( \frac{4}{e^{-\sigma_6 - \sigma_5} + 1} + 15 \right) (x_1 - x_2)$$

$$\sigma_4 = \frac{|x_3|^2}{2}$$

$$\sigma_5 = \frac{|x_2|^2}{2}$$

$$\sigma_6 = \frac{|x_1|^2}{2}$$

## Vincoli su stato e ingresso

Successivamente, si definiscono i vincoli esistenti su stato e ingresso del sistema.

```
x_min_temp = 286.*ones(3,1); % Vincolo di minimo sulla temperatura dell'i-esima stanza
x_min_power = zeros(3,1); % Vincolo di minimo sulla potenza dell'i-esimo termosifone
x_min = [x_min_temp;x_min_power]; % Vincolo di minimo sullo stato
x_max_temp = inf.*ones(3,1); % Vincolo di massimo sulla temperatura dell'i-esima stanza
x_max_power = 150.*ones(3,1); % Vincolo di massimo sulla potenza dell'i-esimo termosifone
x_max = [x_max_temp;x_max_power]; % Vincolo di massimo sullo stato
u_min = zeros(3,1); % Vincolo di minimo sulla potenza dell'i-esimo termosifone
u_max = 150.*ones(3,1); % Vincolo di massimo sulla potenza dell'i-esimo termosifone
```

## Punti di equilibrio

Dalle equazioni sappiamo che il sistema non è lineare, in quanto alcune variabili di stato compaiono nell'esponenziale. Sarà perciò necessario linearizzare il sistema attorno ad un punto di equilibrio. Per calcolare gli ingressi di equilibrio, si azzerava la dinamica del sistema, ponendo a zero le derivate degli stati, e si calcolano le soluzioni del sistema di equazioni sostituendo il punto di equilibrio.

```
% Azzeramento della dinamica
x1d = 0;
x2d = 0;
x3d = 0;
x4d = 0;
x5d = 0;
x6d = 0;

% Sostituzione dei valori noti, ovvero il punto di equilibrio
x1 = x_eq(1);
x2 = x_eq(2);
x3 = x_eq(3);
x4 = x_eq(4);
x5 = x_eq(5);
x6 = x_eq(6);
equazioni_equilibrio = subs(equazioni);

% Ottenimento degli ingressi di equilibrio, u_eq
res = solve(equazioni_equilibrio)
```

```
res = struct with fields:
    u1: 120
    u2: 120
    u3: 120
```

```
u_eq = [double(res.u1) double(res.u2) double(res.u3)]';
```

## Sistema linearizzato

Prima di progettare il regolatore MPC, è necessario linearizzare il sistema attorno al punto di equilibrio calcolato nella sezione precedente, così da calcolare le matrici A e B del sistema linearizzato.

```
syms x1 x2 x3 x4 x5 x6 u1 u2 u3
f1 = (x4 -k12*(x1-x2) -k13*(x1-x3) -k_ext*(x1-T_ext))/C(1);
```

```

f2 = (x5 +k12*(x1-x2) -k23*(x2-x3) -k_ext*(x2-T_ext))/C(2);
f3 = (x6 +k13*(x1-x3) +k23*(x2-x3) -k_ext*(x3-T_ext))/C(3);
f4 = (u1-x4)/tau(1);
f5 = (u2-x5)/tau(2);
f6 = (u3-x6)/tau(3);
sistema = [f1;f2;f3;f4;f5;f6];

```

```

% Calcolo delle derivate parziali rispetto ad ogni stato

```

```

sistema_lin(:,1) = diff(sistema,x1);
sistema_lin(:,2) = diff(sistema,x2);
sistema_lin(:,3) = diff(sistema,x3);
sistema_lin(:,4) = diff(sistema,x4);
sistema_lin(:,5) = diff(sistema,x5);
sistema_lin(:,6) = diff(sistema,x6);
sistema_lin;

```

```

% Calcolo delle derivate parziali rispetto ad ogni ingresso

```

```

ingresso_lin(:,1) = diff(sistema,u1);
ingresso_lin(:,2) = diff(sistema,u2);
ingresso_lin(:,3) = diff(sistema,u3);
ingresso_lin;

```

```

% Sostituzione del punto di equilibrio

```

```

x1 = x_eq(1);
x2 = x_eq(2);
x3 = x_eq(3);
x4 = x_eq(4);
x5 = x_eq(5);
x6 = x_eq(6);
u1 = u_eq(1);
u2 = u_eq(2);
u3 = u_eq(3);
A_ct = subs(sistema_lin)

```

```

A_ct =

```

$$\begin{pmatrix} -\frac{1}{850 (e^{-85849} + 1)} - \frac{43}{6800} & \sigma_2 + \frac{3}{1360} & \sigma_2 + \frac{1}{340} & \frac{1}{6800} & 0 & 0 \\ \sigma_1 & -\frac{1}{700 (e^{-85849} + 1)} - \frac{19}{2800} & \sigma_1 & 0 & \frac{1}{5600} & 0 \\ \frac{1}{\sigma_3} + \frac{1}{255} & \frac{1}{\sigma_3} + \frac{1}{340} & -\frac{2}{\sigma_3} - \frac{43}{5100} & 0 & 0 & \frac{1}{5100} \\ 0 & 0 & 0 & -\frac{1}{400} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{450} & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{450} \end{pmatrix}$$

where

$$\sigma_1 = \frac{1}{1400 (e^{-85849} + 1)} + \frac{3}{1120}$$

$$\sigma_2 = \frac{1}{1700 (e^{-85849} + 1)}$$

$$\sigma_3 = 1275 (e^{-85849} + 1)$$

```
B_ct = subs(ingresso_lin)
```

B\_ct =

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{400} & 0 & 0 \\ 0 & \frac{1}{450} & 0 \\ 0 & 0 & \frac{1}{450} \end{pmatrix}$$

## Autovalori del sistema linearizzato a tempo continuo

Prima di andare a discretizzare il sistema, si analizzano gli autovalori della matrice di stato, A, per studiare la natura del sistema in esame. Questo passaggio è utile sia per capire il comportamento del sistema, che per scegliere il metodo di discretizzazione più adatto.

```
lambda_i = eig(A_ct);
real_lambda_i = double(real(lambda_i))
```

```
real_lambda_i = 6x1
-0.0025
-0.0022
```

```
-0.0022  
-0.0014  
-0.0134  
-0.0110
```

```
imag_lambda_i = double(imag(lambda_i([5,6])))
```

```
imag_lambda_i = 2×1  
10-73 ×  
-0.4643  
0
```

Il sistema ha quattro autovalori reali negativi e una coppia di autovalori complessi coniugati con parte reale negativa, di conseguenza è asintoticamente stabile. Tuttavia, questi valori sono molto prossimi all'asse immaginario, quindi significa che sono molto lenti. La parte immaginaria degli autovalori complessi coniugati è praticamente nulla, ciò significa che la dinamica del sistema non è soggetta ad oscillazioni.

Per progettare il controllore, il sistema deve essere completamente raggiungibile. Verifichiamo quest'ipotesi calcolando il rango della matrice di raggiungibilità del sistema.

```
Mr_ct = ctrb(A_ct,B_ct);  
rk_Mr_ct = rank(Mr_ct)
```

```
rk_Mr_ct = 6
```

Dato che il rango della matrice di raggiungibilità  $M_r$  è pari alla dimensione dello stato,  $n$ , ovvero 6, si può concludere che il sistema è completamente raggiungibile.

## Matrici del sistema linearizzato a tempo discreto

Dato che i controllori lavorano a tempo discreto, è necessario discretizzare le matrici del sistema. Le matrici A e B sono già note a tempo continuo, mentre C e D no. Visto che per il corretto funzionamento del controllore MPC è necessario retroazionare lo stato, si ipotizza che questa misura sia disponibile, ovvero che lo stato sia accessibile, quindi la matrice C è una matrice identità di dimensioni 6x6. Per quanto riguarda la matrice D, sarà una matrice di zeri di dimensioni 6x3, perché si ipotizza che le misure non dipendano dagli ingressi. Una volta ottenuto il sistema a tempo continuo, si ricaverà il sistema a tempo discreto con l'apposito comando di Matlab. Si sceglie di utilizzare il metodo di Tustin, così che il semipiano reale negativo venga mappato nel cerchio di raggio 1 centrato nell'origine. Dato che il sistema è asintoticamente stabile, si poteva utilizzare anche il metodo di Eulero in avanti.

```
C_ct = eye(6);  
D_ct = zeros(6,3);  
continuous_time_ss = ss(double(A_ct),double(B_ct),C_ct,D_ct);  
discrete_time_ss = c2d(continuous_time_ss,Ts,'tustin');  
A = discrete_time_ss.A;  
B = discrete_time_ss.B;
```

Per verificare la correttezza dell'operazione di discretizzazione, si possono analizzare gli autovalori della matrice A e il rango della matrice di raggiungibilità a tempo discreto.

```
lambda_i_dt = eig(A)
```



```
lambda_i_dt = 6x1
    0.9986
    0.9891
    0.9867
    0.9975
    0.9978
    0.9978
```

```
Mr = ctrb(A,B);
rk_Mr = rank(Mr)
```

```
rk_Mr = 6
```

I risultati ottenuti a tempo discreto sono coerenti con quelli ottenuti a tempo continuo: gli autovalori hanno valore molto prossimo all'unità, ciò significa che sono lenti, inoltre il rango della matrice di raggiungibilità rimane pari alla dimensione dello stato, che 6, perciò il sistema è completamente raggiungibile.

## Espressione dei vincoli nelle coordinate del sistema linearizzato

Il progetto del controllore MPC è legato al punto di equilibrio scelto, perché questo viene visto come se fosse l'origine del sistema. Per questo motivo, si devono riesprimere i vincoli e la condizione iniziale nelle coordinate del sistema linearizzato.

```
x_ci_mpc = x_ci-x_eq;
u_eq_mpc = u_eq-u_eq;
x_eq_mpc = zeros(6,1);
x_max_mpc = x_max-x_eq;
x_min_mpc = x_min-x_eq;
u_max_mpc = u_max-u_eq;
u_min_mpc = u_min-u_eq;
```

## Calcolo del Control Invariant Set

Un ingrediente necessario per progettare il controllore MPC è il Control Invariant Set (CIS). Le dimensioni di questo insieme dipendono dalla scelta delle matrici Q ed R del costo quadratico. Il CIS è calcolato iterativamente, ma prima bisogna esprimere i vincoli del sistema nella forma  $H_x x \leq hx$ ,  $H_u u \leq hu$ .

```
% Valori sulla diagonale delle matrici quadrate Q ed R
Q_val = 1;
R_val = 1;
Q = Q_val.*eye(6);
R = R_val.*eye(3);
% Vincoli su stato e ingresso, espressi nella forma di <=
Hx = [eye(6); -eye(6)];
hx = [x_max_mpc;-x_min_mpc];
Hu = [eye(3); -eye(3)];
hu = [u_max_mpc;-u_min_mpc];
% Chiamata alla funzione che calcola il CIS
[G,g] = cis(A,B,x_eq_mpc,u_eq_mpc,Hx,hx,Hu,hu,Q,R);
```

```
ans =
    "iterato 80 volte
    "
```

```
CIS = Polyhedron(G,g);
% Stampa di alcune informazioni riguardanti il CIS appena calcolato
CIS.display()
```

```
Polyhedron in R^6 with representations:
  H-rep (redundant)   : Inequalities 241 | Equalities   0
  V-rep              : Unknown (call computeVRep() to compute)
Functions : none
```

```
CIS.isBounded()
```

```
ans = logical
      1
```

```
CIS.isEmptySet()
```

```
ans = logical
      0
```

```
CIS.isFullDim()
```

```
ans = logical
      1
```

Dato che non è possibile visualizzare il grafico del poliedro, in quanto appartenente allo spazio 6D, si utilizzano delle funzioni di MPT3 per ricavare alcune proprietà di questo poliedro. Alcuni vincoli sono ridondanti, la regione individuata non è vuota ed è limitata, inoltre ha dimensione pari alla dimensione dello stato, che è 6.

## Formulazione del problema quadratico

...

```
% Orizzonte di predizione
N = 10;

nx = size(A,1);
nu = size(B,2);

% u minima, massima e iniziale per quadprog
u_min_qp = zeros(nu*N,1);
u_max_qp = zeros(nu*N,1);
u_ci_qp = zeros(nu*N,1);
u_ci_mpc = [x_ci_mpc(4);x_ci_mpc(5);x_ci_mpc(6)];
for i = 0:1:N-1
    if(i==0)
        u_min_qp = u_min_mpc;
        u_max_qp = u_max_mpc;
        u_ci_qp = u_ci_mpc;
    else
        u_min_qp = [u_min_qp;u_min_mpc];
        u_max_qp = [u_max_qp;u_max_mpc];
        u_ci_qp = [u_ci_qp;u_ci_mpc];
    end
end
```

## % Controllo aggressivo

```
Q_val = 10;
R_val = 0.1;
Q_hard = Q_val.*Q;
R_hard = R_val.*R;

[H_hard,f_hard,Aineq_hard,bineq_hard] = quadprogmatrix(N,nx,nu,A,B,Q_hard,R_hard,x_ci,Hx,hx,Hu,
```

```
H = 30×30
    0.1027    0.0000    0.0000    0.0027    0.0000    0.0000    0.0026    0.0000 ...
    0.0000    0.1024    0.0000    0.0000    0.0023    0.0000    0.0000    0.0023
    0.0000    0.0000    0.1024    0.0000    0.0000    0.0023    0.0000    0.0000
    0.0027    0.0000    0.0000    0.1027    0.0000    0.0000    0.0026    0.0000
    0.0000    0.0023    0.0000    0.0000    0.1023    0.0000    0.0000    0.0023
    0.0000    0.0000    0.0023    0.0000    0.0000    0.1023    0.0000    0.0000
    0.0026    0.0000    0.0000    0.0026    0.0000    0.0000    0.1026    0.0000
    0.0000    0.0023    0.0000    0.0000    0.0023    0.0000    0.0000    0.1023
    0.0000    0.0000    0.0023    0.0000    0.0000    0.0023    0.0000    0.0000
    0.0026    0.0000    0.0000    0.0026    0.0000    0.0000    0.0026    0.0000
    ⋮
```

```
f = 30×1
    18.8286
    18.6644
    18.7621
    18.4633
    18.3328
    18.4287
    18.0977
    18.0010
    18.0952
    17.7318
    ⋮
```

```
H_hard = 30×30
    0.1027    0.0000    0.0000    0.0027    0.0000    0.0000    0.0026    0.0000 ...
    0.0000    0.1024    0.0000    0.0000    0.0023    0.0000    0.0000    0.0023
    0.0000    0.0000    0.1024    0.0000    0.0000    0.0023    0.0000    0.0000
    0.0027    0.0000    0.0000    0.1027    0.0000    0.0000    0.0026    0.0000
    0.0000    0.0023    0.0000    0.0000    0.1023    0.0000    0.0000    0.0023
    0.0000    0.0000    0.0023    0.0000    0.0000    0.1023    0.0000    0.0000
    0.0026    0.0000    0.0000    0.0026    0.0000    0.0000    0.1026    0.0000
    0.0000    0.0023    0.0000    0.0000    0.0023    0.0000    0.0000    0.1023
    0.0000    0.0000    0.0023    0.0000    0.0000    0.0023    0.0000    0.0000
    0.0026    0.0000    0.0000    0.0026    0.0000    0.0000    0.0026    0.0000
    ⋮
```

```
f_hard = 30×1
    18.8286
    18.6644
    18.7621
    18.4633
    18.3328
    18.4287
    18.0977
    18.0010
    18.0952
    17.7318
    ⋮
```

```
Aineq_hard = 192×30
    1     0     0     0     0     0     0     0     0     0     0     0     0 ...
    0     1     0     0     0     0     0     0     0     0     0     0     0
```

```

0      0      1      0      0      0      0      0      0      0      0      0      0
-1     0      0      0      0      0      0      0      0      0      0      0      0
0     -1     0      0      0      0      0      0      0      0      0      0      0
0      0     -1     0      0      0      0      0      0      0      0      0      0
0      0      0      1      0      0      0      0      0      0      0      0      0
0      0      0      0      1      0      0      0      0      0      0      0      0
0      0      0      0      0      1      0      0      0      0      0      0      0
0      0      0     -1      0      0      0      0      0      0      0      0      0
:
:
bineq_hard = 192x1
30
30
30
120
120
120
30
30
30
120
:
:

```

```
u = quadprog(H_hard,f_hard,Aineq_hard,bineq_hard,[],[],u_min_qp,u_max_qp,u_ci_qp)
```

The interior-point-convex algorithm does not accept an initial point.  
Ignoring X0.

Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in  
feasible directions, to within the value of the optimality tolerance,  
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>

```

u = 30x1
-120.0000
-120.0000
-120.0000
-120.0000
-120.0000
-120.0000
-120.0000
-120.0000
-120.0000
-120.0000
-120.0000
:
:

```

```

u_shifted_hard = zeros(30,1);
for i = 0:1:N-1
    if(i==0)
        u_shifted_hard(1) = u(1)+u_eq(1);
        u_shifted_hard(2) = u(2)+u_eq(2);
        u_shifted_hard(3) = u(3)+u_eq(3);
    else
        u_shifted_hard(i*nu+1) = u(i*nu+1)+u_eq(1);
        u_shifted_hard(i*nu+2) = u(i*nu+2)+u_eq(2);
        u_shifted_hard(i*nu+3) = u(i*nu+3)+u_eq(3);
    end
end

```

```

end
end
u_shifted_hard

```

```

u_shifted_hard = 30x1
10^-8 x
    0.0212
    0.0230
    0.0241
    0.0194
    0.0211
    0.0211
    0.0058
    0.0080
    0.0155
    0.0741
    ⋮

```

```

% % Controllo delicato
% Q_val = 1;
% R_val = 10;
% Q_soft = Q_val.*Q;
% R_soft = R_val.*R;
%
% [H_soft,f_soft,Aineq_soft,bineq_soft] = quadprogmatrix(N,nx,nu,A,B,Q_soft,R_soft,x_ci,Hx,hx,h)
%
% u = quadprog(H_hard,f_hard,Aineq_hard,bineq_hard,[],[],u_min_qp,u_max_qp,u_ci_qp)
%
% u_shifted_soft = zeros(30,1);
% for i = 0:1:N-1
%     if(i==0)
%         u_shifted_soft(1) = u(1)+u_eq(1);
%         u_shifted_soft(2) = u(2)+u_eq(2);
%         u_shifted_soft(3) = u(3)+u_eq(3);
%     else
%         u_shifted_soft(i*nu+1) = u(i*nu+1)+u_eq(1);
%         u_shifted_soft(i*nu+2) = u(i*nu+2)+u_eq(2);
%         u_shifted_soft(i*nu+3) = u(i*nu+3)+u_eq(3);
%     end
% end
% u_shifted_soft

```

## Formulazione del controllore MPC

...

## Analisi delle prestazioni del controllore MPC progettato

...

## Funzioni

Di seguito, si riporta la definizione delle funzioni utilizzate nelle sezioni precedenti.

### 1. Calcolo del Control Invariant Set

```
function [G,g] = cis(A,B,x_ref,u_ref,Fx,fx,Fu,fu,Q,R)
% Controllore LQR
K = -dlqr(A,B,Q,R); % Segno - per usare con + sotto
% Matrice A del sistema controllato con LQR
A_lqr = A+B*K;
% Vincoli su stato e ingresso
F = [Fx; Fu*K];
f = [fx; fu + Fu*(K*x_ref - u_ref)];
% Calcolo CIS ( $G^*x \leq g$ )
% Inizializzazione
CIS_poly_prev = Polyhedron(); % Poliedro vuoto
CIS_poly_curr = Polyhedron(F,f); % Poliedro fornito dai vincoli appena trovati
% La prima condizione del ciclo while (CIS_poly_prev.isEmptySet) serve per
% non uscire alla prima iterazione
i=1;
while CIS_poly_prev.isEmptySet || CIS_poly_prev ~= CIS_poly_curr
    % memorizzo il vecchio set candidato
    CIS_poly_prev = CIS_poly_curr;
    % Calcolo il nuovo set candidato ( $G_{hat}x \leq g_{hat}$ )
    G_hat = [CIS_poly_curr.A * A_lqr; F]; % tutti i vincoli stato
    g_hat = [CIS_poly_prev.b + CIS_poly_curr.A*B*(K*x_ref - u_ref); f];
    CIS_poly_curr = Polyhedron(G_hat,g_hat); % Poliedro con nuovi vincoli
    i=i+1;
end
sprintf("iterato %d volte \n",i)
G = CIS_poly_curr.A;
g = CIS_poly_curr.b;
end
```

### 2. Calcolo delle matrici richieste da quadrog

```
function [H,f,Aineq,bineq] = quadprogmatrix(N,nx,nu,A,B,Q,R,x_ci,Hx,hx,Hu,hu)
A_italic = zeros(nx*(N+1),nx);
%% B_italic = zeros(nx*(N+1),nu*N);
for i = 0:1:N
    if(i==0)
        A_italic = eye(nx);
        %% B_italic_prova(1:(nx),:) = eye(nx,nu*N);
    else
        A_italic = [A_italic;A.^i];
        %% B_italic_prova(1:(i*nx)+nx,:) = [B_italic_prova;A^(i-1)*B MANCA];
    end
end

B_italic = [zeros(nx,nu*N);
    B zeros(nx,nu*(N-1));
    A^(N-9)*B B zeros(nx,nu*(N-2));
    A^(N-8)*B A^(N-9)*B B zeros(nx,nu*(N-3));
```

```

A^(N-7)*B A^(N-8)*B A^(N-9)*B B zeros(nx,nu*(N-4));
A^(N-6)*B A^(N-7)*B A^(N-8)*B A^(N-9)*B B zeros(nx,nu*(N-5));
A^(N-5)*B A^(N-6)*B A^(N-7)*B A^(N-8)*B A^(N-9)*B B zeros(nx,nu*(N-6));
A^(N-4)*B A^(N-5)*B A^(N-6)*B A^(N-7)*B A^(N-8)*B A^(N-9)*B B zeros(nx,nu*(N-7));
A^(N-3)*B A^(N-4)*B A^(N-5)*B A^(N-6)*B A^(N-7)*B A^(N-8)*B A^(N-9)*B B zeros(nx,nu*(N-8));
A^(N-2)*B A^(N-3)*B A^(N-4)*B A^(N-5)*B A^(N-6)*B A^(N-7)*B A^(N-8)*B A^(N-9)*B B zeros(nx,nu*(N-9));
A^(N-1)*B A^(N-2)*B A^(N-3)*B A^(N-4)*B A^(N-5)*B A^(N-6)*B A^(N-7)*B A^(N-8)*B A^(N-9)*B B zeros(nx,nu*(N-10));
%% FINIRE DI MODIFICARE B_italic COME PARAMETRICA

[K,P] = dlqr(A,B,Q,R);
Q_italic = zeros(nx*(N+1),nx*(N+1));
R_italic = zeros(nu*N,nu*N);
for i = 0:1:N-1
    Q_italic((i*nx)+1:(i*nx)+nx,(i*nx)+1:(i*nx)+nx) = diag(diag(Q));
    R_italic((i*nu)+1:(i*nu)+nu,(i*nu)+1:(i*nu)+nu) = diag(diag(R));
end
Q_italic((N*nx)+1:(N*nx)+nx,(N*nx)+1:(N*nx)+nx) = P;

prod = B_italic' * Q_italic * B_italic;

for i=1:length(prod)
    for j=1:length(prod)
        prod(i,j)=round(prod(i,j),5,"significant");
    end
end

H = R_italic + prod %% --> risulta 30x30

f =(x_ci' * A_italic' * Q_italic * B_italic)' %% --> risulta 30x1 + SOSTITUIRE CI CON Z

% Calcolo matrici dei vincoli sugli ingressi
Hu_italic = zeros(2*nu*N,N*nu);
hu_italic = zeros(2*nu*N,1);
for i = 0:1:N-1
    if(i==0)
        hu_italic = hu;
        Hu_italic(1:2*nu,:) = [Hu zeros(2*nu,(N-1)*nu)];
    else
        hu_italic = [hu_italic;hu];
        if(i==N-1)
            Hu_italic((i*2*nu)+1:(i*2*nu)+(2*nu),:) = [zeros(2*nu,i*nu) Hu];
        else
            Hu_italic((i*2*nu)+1:(i*2*nu)+(2*nu),:) = [zeros(2*nu,i*nu) Hu zeros(2*nu,(N-i-1)*nu)];
        end
    end
end

% Calcolo matrici dei vincoli sullo stato
Hx_italic = zeros(2*nx*(N+1),(N+1)*nx);
hx_italic = zeros(2*nx*(N+1),1);
for i = 0:1:N
    if(i==0)
        hx_italic = hx;
        Hx_italic(1:2*nx,:) = [Hx zeros(2*nx,N*nx)];
    end
end

```

```

else
    hx_italic = [hx_italic;hx];
    if(i==N)
        Hx_italic((i*2*nx)+1:(i*2*nx)+(2*nx),:) = [zeros(2*nx,N*nx) Hx];
    else
        Hx_italic((i*2*nx)+1:(i*2*nx)+(2*nx),:) = [zeros(2*nx,i*nx) Hx zeros(2*nx,(N-i)*nx)];
    end
end

end

Aineq = [Hu_italic; Hx_italic * B_italic];
bineq = [hu_italic; hx_italic - Hx_italic * A_italic * x_ci]; %% SOSTITUIRE CI CON Z
end

```