Esercitazione di Controlli Automatici
"Dalle equazioni non lineari di un sistema alla pianificazione ottima e utilizzo con il controllo"

18 dicembre 2014

1 Descrizione del Problema

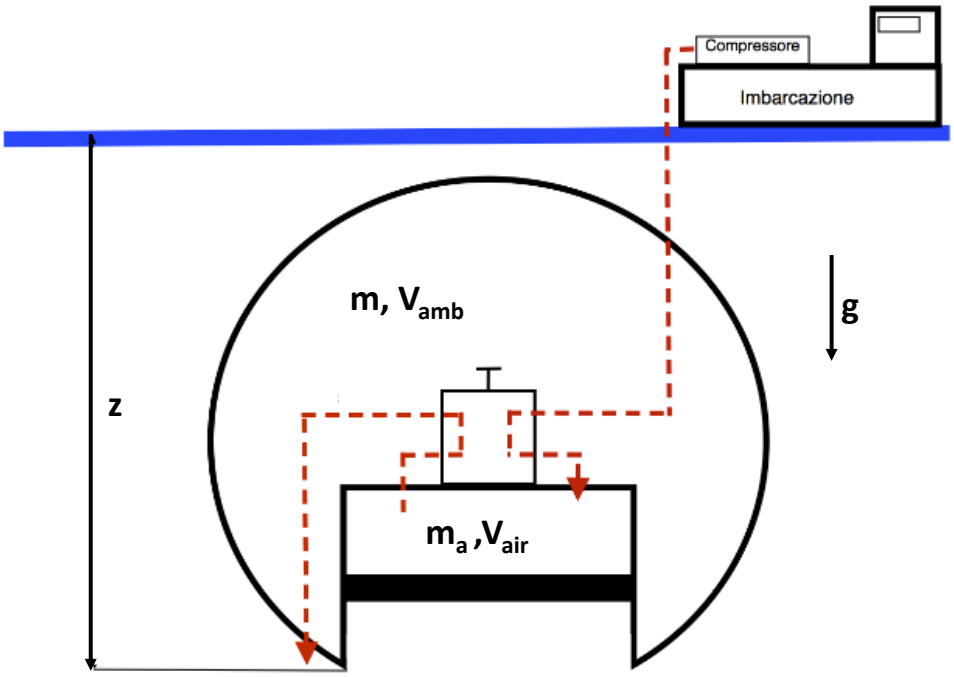

Figura 1: Schema semplificato di un piccolo corpo sommergibile comandato con camera di compensazione (elaborato da M. Barbarossa)

Si consideri la dinamica semplificata di un piccolo sommergibile di massa m nel moto di immersione ed emersione. Il sommergibile è costituito da una struttura sferica di raggio r (figura 1) composta da due ambienti, uno rigido abitabile di volume V_{amb} ed una camera di compensazione. Il controllo avviene regolando la portata di aria (fornita a pressione costante da un compressore posto su di un'imbarcazione in superficie) entrante e uscente dalla cassa attraverso una valvola di apertura v: detta m_a la massa d'aria nel sommergibile, variando quest'ultima si varia il volume della cassa di compensazione e quindi la spinta di Archimede complessiva.

Detta z la profondità d'immersione del sommergibile positiva nel verso della discesa e modellando possibili disturbi quali correnti come una forza F agente sul sommergibile, le equazioni che descrivono la dinamica del sistema (accelerazione verticale e variazione della massa d'aria) sono le seguenti:

$$\ddot{z} = -\frac{b\,\dot{z}}{m} + \alpha - \frac{\beta\, m_a}{p_0 + \rho_H\, g\, z} - \frac{F}{m}$$
$$\dot{m}_a = (p_0 + \rho_H\, g\, z)\,\gamma\, v$$

dove b è lo smorzamento viscoso dell'acqua sul sommergibile, p_0 la pressione atmosferica a livello del mare ($z = 0$), ρ_H la densità dell'acqua, e le costanti α, β e γ (riportate solo per completezza) sono definite come

$$\alpha = \frac{g}{m}(m - \rho_H V_{\text{amb}}) \qquad \beta = \frac{\rho_H\, g\, R_a\, T_a}{m} \qquad \gamma = \frac{\dot{V}_{\max}}{R_a\, T_a}$$

in cui R_a è la costante caratteristica dell'aria, T_a la temperatura ambiente e $\dot{V}_{\max}$ la massima portata volumetrica fra il compressore ed il sommergibile.

A.1 Si determinino tutti gli equilibri fisicamente ammissibili del sistema per disturbo assente.

A.2 Supponendo di disporre della misura della posizione z, si determini una rappresentazione in forma di stato simbolica del sistema linearizzato intorno ad un equilibrio come calcolato al punto precedente per un valore della profondità $z = \bar{z}$.

Successivamente, si considerino i seguenti valori numerici:

$$m = 6 \cdot 10^3\ [Kg]\,,\ g = 9.81\ \left[m/s^2\right]\,,\ b = 0.03013\,[N\,s/m],\ \alpha = 0.981\,[m/s^2],\ \beta = 1.329 \cdot 10^5\,[s^{-4}]$$

$$\gamma = 2.45 \cdot 10^{-6}\,[m\,s],\ \rho_H = 10^3\ \left[Kg/m^3\right]\,,\ p_0 = 10^5\ [Pa]\,,\ \bar{z} = 300\ [m]$$

A.3 Si progetti un compensatore basato su regolatore che stabilizzi il sistema.

A.4 Si determini una legge di controllo prepianificata che raggiunga lo stato finale desiderato ad un altezza $z = \bar{z} + 20$, calcolando opportunamente il valore della restante parte dello stato e dell'ingresso in modo tale da poter raggiungere e rimanere indefinitamente in tale posizione; si consideri a tal fine una opportuna discretizzazione del sistema in esame ed un ingresso costante a tratti di durata pari al tempo di campionamento T_s. Fare ciò in modo incrementale, considerando di

A.4.1 non avere alcun vincolo, voler utilizzare il tempo minimo possibile, ottimizzando la norma 2 del vettore di controllo

A.4.2 avere una variabile di attuazione limitata in $v \in [-20 \div 20]$, sempre minimizzando la norma 2 del vettore di controllo ed il tempo minimo possibile a T_s fissato

A.4.3 utilizzare un tempo pari al precedente più 1 sec, minimizzando stavolta una funzione di costo arbitraria *myFun()*, scegliendo per esempio di minimizzare la massima variazione dell'ingresso in passi adiacenti

A.4.4 aggiungere un vincolo sullo stato in modo che la variabile $\dot{z}$ non esca mai dall'intervallo $\{-6 \div 6\}$ m/s

A.5 Si effettui una simulazione del sistema con ingresso ottimo come al punto **A.4.4** e controllore ottenuto al punto **A.3** in retroazione

A.5.1 sul sistema lineare

A.5.1 sul sistema non lineare

2 Soluzione

A.1 Indicando con $x = [z\ \dot{z}\ m_{air}]^{\mathrm{T}}$ il vettore di stato ed imponendo le condizioni di equilibrio, $\dot{x} = 0$ con $F = 0$ si ottiene:

$$\begin{aligned} 0 &= \alpha - \frac{\beta\,\bar{m}_a}{p_0 + \rho_H\, g\,\bar{z}} \\ 0 &= (p_0 + \rho_H\, g\,\bar{z})\,\gamma\,\bar{v}\,. \end{aligned}$$

Dalla prima equazione si ottiene l'espressione della massa d'aria $\bar{m}_a$ in funzione di $\bar{z}$

$$\bar{m}_a = \frac{\alpha}{\beta}\,(p_0 + \rho_H\, g\,\bar{z}) \tag{1}$$

mentre dalla seconda, essendo questa una moltiplicazione, si ricavano due diverse condizioni:

- $\bar{v} = 0$
- $\bar{z} = -\frac{p_0}{\rho_H\, g}$

dato che il rapporto contiene solo quantità positive, la seconda condizione richiede che sia $z < 0$, ovvero una posizione al di sopra del livello del mare. Dunque l'unica soluzione che ha senso fisico è quella con $\bar{v} = 0$ e $\bar{z} > 0$ qualsiasi.

A.2 Indicando con $\tilde{x} = [\tilde{x}_1,\ \tilde{x}_2,\ \tilde{x}_3]^T = [x_1 - \bar{z},\ x_2,\ x_3 - \bar{m}_a]^T$, il vettore delle variabili di stato traslate nell'equilibrio e con $\tilde{u} = [\tilde{u}_1,\ \tilde{u}_2]^T = [u - \bar{u},\ W]^T$ il vettore degli ingressi anch'essi traslati, il sistema non lineare scritto in forma di stato traslato attorno all'equilibrio è

$$\begin{cases} \dot{\tilde{x}}_1 = \tilde{x}_2 \\ \dot{\tilde{x}}_2 = -\frac{b}{m}\,\tilde{x}_2 + \alpha - \beta\dfrac{\tilde{x}_3 + \bar{m}_a}{p_0 + \rho_H\, g\,(\tilde{x}_1 + \bar{z})} - \dfrac{\tilde{u}_2}{m} \\ \dot{\tilde{x}}_3 = (p_0 + \rho_H\, g\,(\tilde{x}_1 + \bar{z}))\,\gamma\,(\tilde{u}_1 + \bar{v}) \end{cases}$$

Linearizzando il sistema attorno all'origine, essendo questo l'equilibrio delle nuove variabili $\tilde{x}$ e $\tilde{u}$, si ottiene il sistema linearizzato approssimato nella consueta forma di stato

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{2}$$

dove,

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \dfrac{\beta\, g\, \rho_H\, \bar{m}_a}{(p_0 + g\,\rho_H\,\bar{z})^2} & -\dfrac{b}{m} & -\dfrac{\beta}{p_0 + g\,\rho_H\,\bar{z}} \\ g\,\gamma\,\rho_H \bar{v} & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ \dfrac{\alpha\, g\, \rho_H}{p_0 + g\,\rho_H\,\bar{z}} & -\dfrac{b}{m} & -\dfrac{\beta}{p_0 + g\,\rho_H\,\bar{z}} \\ 0 & 0 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & -\dfrac{1}{m} \\ \gamma\,(p_0 + g\,\rho_H\,\bar{z}) & 0 \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix},$$

$$D = \begin{bmatrix} 0 & 0 \end{bmatrix}.$$

Sostituendo i valori numerici assegnati si ottiene:

$$\begin{aligned} A &= \begin{bmatrix} 0 & 1 & 0 \\ 0.0032 & -5 \cdot 10^{-6} & -0.0437 \\ 0 & 0 & 0 \end{bmatrix}, \\ B &= \begin{bmatrix} 0 & 0 \\ 0 & -0.0002 \\ 7.4858 & 0 \end{bmatrix}, \\ C &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \\ D &= \begin{bmatrix} 0 & 0 \end{bmatrix}. \end{aligned}$$

da cui i poli del sistema risultano essere $[0, -0.056, 0.056]$: uno di questi poli è a parte reale positiva ed il sistema è dunque instabile. Questo si può interpretare da un punto di vista fisico col fatto che se ci si trova in condizioni diverse da quelle di equilibrio, ad esempio ad una profondità inferiore, la pressione a tale profondità sarà minore e conseguentemente il volume della cassa di compensazione maggiore, avendo una maggiore spinta di Archimede ed una tendenza a spostarsi verso l'alto e divergendo. Stesse considerazioni, all'opposto, si possono fare per il caso di profondità superiori a quella di equilibrio.

A.3 Per progettare un compensatore basato su regolatore si deve verificare che il sistema *sys* sia completamente raggiungibile dal solo ingresso di controllo $\tilde{u}_1$ e completamente osservabile dal solo ingresso di misura $\tilde{x}_1$, controllando che le matrici di raggiungibilità R e osservabilità O siano di rango pieno:

```
% osservabilita' numerica
rO = rank(obsv(sys.a,sys.c));
disp(['Il rango della matrice di osservabilita'' con uscita x1 e'' ' ...
    num2str(rO)]);

% raggiungibilita' numerica
rR = rank(ctrb(sys.a,sys.b));
disp(['Il rango della matrice di raggiungibilita'' con ingresso u1 e'' ' ...
    num2str(rR)]);
```

Dopodiché, nel caso in cui queste due condizioni siano verificate, si può passare ad assegnare i poli del sistema controllato in ciclo chiuso attraverso una retroazione statica degli stati con matrice dei guadagni K, ed a costruire un osservatore asintotico dello stato attraverso una matrice di iniezione delle uscite L. Questo può essere fatto in Matlab utilizzando i seguenti comandi

```
% choose close-loop system poles, e.g. -1,-2,..-n
p = (-1:-1:-n).';
% pole placement
K = place(sys.a,sys.b,p);

% observer poles (faster then controller)
q = 5*p;
% pole placement
L = (place(sys.a.',sys.c.',q)).';

% regulator:
%   Ar = A-B_u*K-L*C
%   Br = L
%   Cr = K
reg = ss( sys.a-sys.b*K-L*sys.c, L, K, 0 );
```

Si può vedere come il regolatore sia stato sintetizzato e le sue matrici siano contenute nella variabile di sistema in spazio di stato `reg`, ovvero siano `reg.a`, `reg.b`, `reg.c` e `reg.d`.
Per verificare che il sistema si comporti effettivamente in modo desiderato (specie se ci sono particolari performance richieste), è utile controllare la risposta a gradino del sistema in ciclo chiuso con il controllore

```
Gcl=feedback(sys,reg);
opt=stepDataOptions('StepAmplitude', dcgain(rsys));
step(Gcl,opt);
```

da cui si vede la risposta a gradino di Figura 2

La stessa cosa può essere vista utilizzando uno schema Simulink come quello di Figura 3. Si può osservare da Figura 4 che le due coincidono.
Nota: nel blocco `To Workspace`, inserire il nome della variabile che si desidera salvare nello spazio di lavoro con la simulazione, ad esempio `yOut`, e selezionare come tipo di dato `Structure with Time`; utilizzare poi i seguenti comandi per il plot

```
plot(yOut.time,yOut.signals.values)
xlabel('Time (seconds)');
```

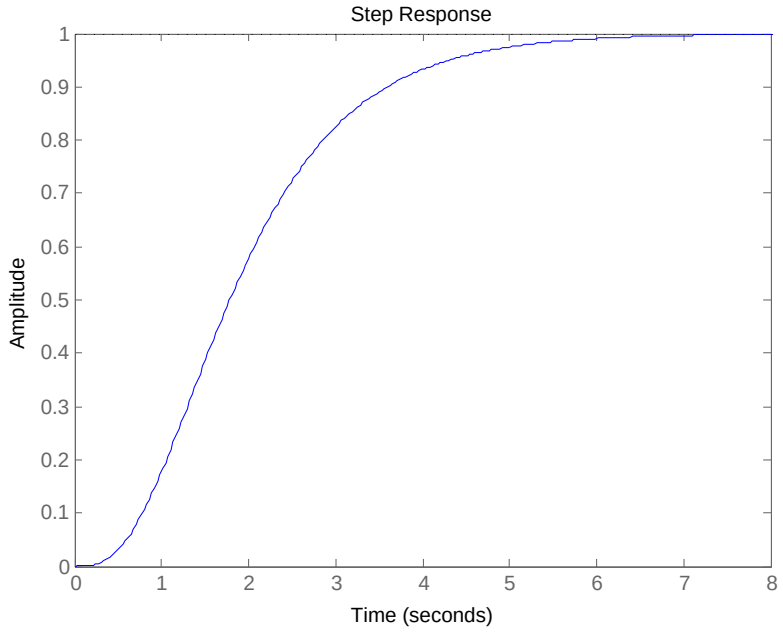

Figura 2: Risposta al gradino del sistema ottenuta con il comando `step`.

```
ylabel('Amplitude');
title('Step Response');
```

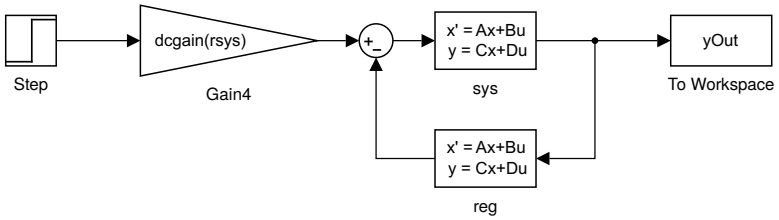

Figura 3: Schema Simulink del sistema lineare con regolatore in retroazione per simulare una risposta a gradino.

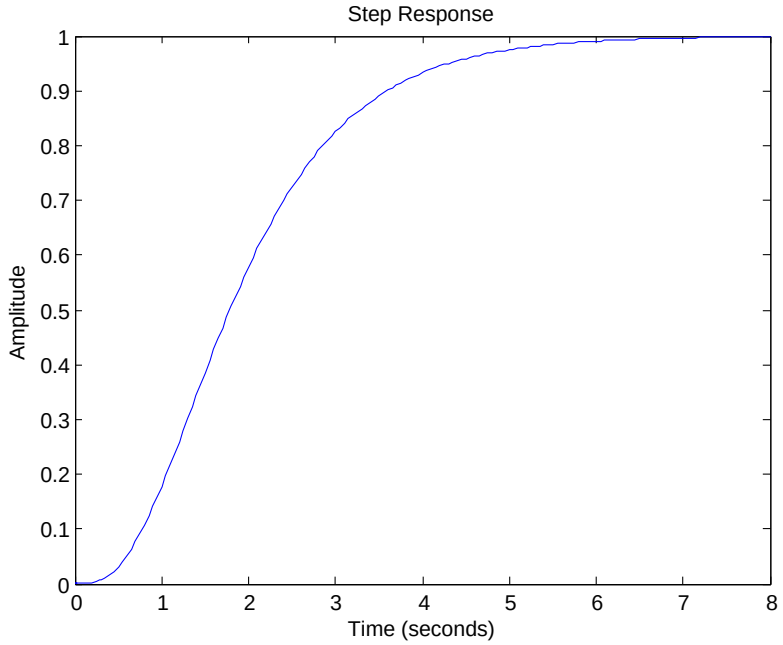

Figura 4: Risposta a gradino ottenuta attraverso una simulazione del sistema tramite Simulink.

Si può notare come in entrambi i casi sia necessario moltiplicare il riferimento a gradino per il guadagno statico del regolatore (ottenuto con l'istruzione `dcgain(rsys)`), così da avere un guadagno statico complessivo unitario.

Un'ulteriore considerazione: la scelta dei poli nell'esempio è stata semplicemente di porre quelli della matrice $A - B\,K$ del sistema controllato in $[-1, -2, -3]$ e quelli dell'osservatore, 5 volte più veloci, in $[-5, -10, -15]$.
Se si hanno delle specifiche da far rispettare al sistema, la posizione dei poli del ciclo chiuso dovrà essere tale da far sì che queste siano verificate. Volendo ad esempio che l'assestamento al 5% di una risposta a gradino sia non superiore a 3 sec, si ottengono dalle condizioni sulle approssimazioni del sistema in ciclo chiuso con un primo o con un secondo ordine le relazioni

$$\omega_n \geq \frac{3}{T_{a,\,5}} \tag{3}$$

per un primo ordine e

$$\delta\,\omega_n \geq \frac{3}{T_{a,\,5}} \tag{4}$$

per un secondo ordine. Entrambe corrispondono a richiedere una parte reale dei poli dominanti superiore in modulo a $\frac{3}{T_a} = 1$.
Nel caso in cui fosse necessario inserire uno o più poli nell'origine per soddisfare specifiche di reiezione di disturbi o di errore a regime, questo deve essere fatto attraverso una estensione del sistema originale ponendovi un numero adeguato di integratori in serie ed effettuando il progetto del regolatore sul sistema esteso: si consideri ad esempio il seguente codice Matlab per l'inserimento di un integratore ed il successivo progetto del regolatore

```
% choose closed-loop system poles
p = ... ;

% insert an integrator in series to the system
integrator = 1/tf('s');
% extended system
sys_e = integrator*sys;

% pole placement
K = place(sys_e.a,sys_e.b,p);
% observer poles (faster then controller)
q = 5*p;
% pole placement
L = place(sys_e.a.',sys_e.c.',q).';

% regulator (another way to obtain the same system)
rsys = -reg(sys_e,K,L);
```

Una scelta diversa delle matrici K ed L: nel caso in cui sia possibile (o preferibile) esprimere i requisiti sul sistema in ciclo chiuso come un indice di prestazione da minimizzare del tipo

$$J_{\mathrm{LQR}} = \int_0^\infty \left(x^{\mathrm{T}}\,Q\,x + u^{\mathrm{T}}\,R\,u\right)\,dt \tag{5}$$

dove Q ed R sono matrici quadrate di dimensione opportuna che pesano il contributo dello stato e dell'ingresso nell'indice di costo, una tecnica per la scelta della matrice di retroazione K è la tecnica *Linear-Quadratic Regulator* (LQR). Nello specifico caso di un sistema SISO in cui si ha interesse ad imporre una performance sull'uscita $y = C\,x$, si possono usare le matrici

$$Q = C^{\mathrm{T}}\,C \in \mathbb{R}^{n\times n}, \quad R = \rho \in \mathbb{R} \tag{6}$$

ed usare il valore ρ come unico parametro di *tuning* per scegliere l'opportuno trade-off fra performance sull'uscita e sforzo di controllo.
Un approccio simile si può utilizzare per scegliere la matrice di iniezione delle uscite L una volta che sia noto il trade-off fra bontà del modello ed accuratezza della misurazione delle uscite (espresso in questa

formulazione come covarianza dei rumori di processo e di misura): si utilizza in questo caso la procedura di sintesi che restituisce il cosiddetto *Filtro di Kalman*.

Per una visione lievemente più completa delle potenzialità di questi strumenti si rimanda alla letteratura e all'help dei comandi Matlab `lqr` e `kalman`.

A.4 Per prima cosa, si deve passare da una realizzazione continua del sistema ad una sua approssimazione discreta. Per la scelta fatta sugli ingressi costanti a tratti, è conveniente utilizzare il metodo di discretizzazione Zero-Order Hold (ZOH), che produce il sistema tempo-discreto

$$\begin{aligned} \xi_{k+1} &= A_Z\,\xi_k + B_Z\,u_k \\ y_k &= C_Z\,\xi_k\,, \end{aligned}$$

coincidente con il sistema tempo-continuo ad ogni istante di campionamento.
Mentre si ha $C_Z = C$ (la matrice di uscita nel caso tempo-discreto è uguale al sistema tempo-continuo), le matrici A_Z e B_Z del sistema tempo discreto sono date dall'uguaglianza

$$e^{\begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} T_s} = \begin{bmatrix} A_Z & B_Z \\ 0 & I \end{bmatrix}$$

dove gli zeri nella matrice che appare nell'esponenziale sono di dimensioni opportune per rendere la matrice stessa quadrata. Le stesse matrici possono essere ottenute mediante il comando Matlab

```
sysD = c2d(sys,Ts,'zoh');
Ad = sysD.a;
Bd = sysD.b;
```

dove `sys` è il sistema tempo continuo in forma di stato mentre `Ts` è il tempo di campionamento.
Lo stesso risultato può essere ottenuto, sempre in Matlab, utilizzando i comandi

```
n = size(A,1);
m = size(B,2);
M = expm([A B; zeros(m,n+m)]*Ts);
Ad = M(1:n,1:n);
Bd = M(1:n,n+1:n+m);
```

La scelta del tempo di campionamento T_s può essere fatta considerando le costanti di tempo dei poli del sistema e scegliendo un tempo almeno $5 \div 10$ volte inferiore. Nell'esempio si è scelto $T_s = 0.05$ sec.

Si noti che la proprietà di raggiungibilità di questo sistema deve essere nuovamente verificata, essendo tale proprietà non garantita nell'approssimazione da tempo continuo a tempo discreto.

```
Rd = ctrb(Ad,Bd);
if (rank(Rd) < n)
    warning('The discrete system is NOT completely reachable!!!');
else
    disp('Discrete system completely reachable');
end
```

Si procede adesso al calcolo della coppia stato-ingresso di equilibrio che ci porta in una posizione $z = \bar{z}+20$. Per quanto riguarda lo stato, nel sistema nonlineare di partenza sappiamo che la forma degli equilibri è data da (1), e quindi per $z_f = \bar{z}+20$ m, ovvero per $x_{f,1} = 20$ m, il valore di $x_{f,3}$ può essere calcolato come

$$x_{f,3} = \frac{\alpha}{\beta}\,(p_0 + \rho_H\,g\,z_f) - \frac{\alpha}{\beta}\,(p_0 + \rho_H\,g\,\bar{z}) = \frac{\alpha}{\beta}\rho_H\,g\,x_{f,1}\,, \tag{7}$$

mentre $x_{f,2} = 0$ per potersi mantenere fermi (velocità di discesa nulla).
Il valore di controllo di equilibrio, come visto al punto **A.1**, è pari a zero. In generale, si può calcolare tale valore per poterlo dare costantemente dal passo $p+1$ in poi (p sono gli istanti usati nella pianificazione) come

```
% last control value
up = pinv(B_u)*(-A*xf);
```

che restituisce sicuramente un qualche valore; per essere sicuri che l'ingresso sia anche un ingresso di equilibrio, si può testare che la derivata dello stato sia pari a zero, a meno di una certa tolleranza usata per non risentire di eventuali errori numerici (e.g. 10^{-10})

```
% check whether up is an equilibrium control value
if (norm(A*xf+B_u*up) < 1e-10)
    disp('Equilibrium control correctly found')
else
    warning('u(p) is NOT an equilibrium control!!!')
end
```

A.4.1 minima norma 2 - nessun vincolo - tempo minimo $T_{\min,1}$

Per raggiungere un determinato stato, avendo controllato che anche il sistema discreto è completamente raggiungibile, servono in generale al più un numero di passi pari alla dimensione dello stato (in questo caso $p = 3$).
Esplicitando il vincolo di raggiungimento dell'obiettivo in p passi si scrive

$$\xi_p = A_Z^p \, \xi_0 + R_p U_p = \xi_f$$

dove R_p è la matrice di raggiungibilità in p passi, ovvero

$$R_p = \begin{bmatrix} B_Z & A_Z \, B_Z & A_Z^2 \, B_Z & \cdots & A_Z^{N-2} \, B_Z & A_Z^{N-1} \, B_Z \end{bmatrix} \tag{8}$$

e dove il vettore

$$U_p = \begin{bmatrix} u_{p-1} \\ \vdots \\ u_0 \end{bmatrix}$$

impila l'intera sequenza dei valori di controllo, in ordine inverso.

Il problema di controllo ottimo che minimizza la norma 2 del vettore degli ingressi può essere posto nei seguenti termini:

$$\begin{cases} \widehat{U}_p = \arg\min_{U_p} U_p^{\mathrm{T}} \, U_p \\ \\ \text{soggetto a} \quad : \\ \qquad \xi_f \quad = \quad A_Z^N \, \xi_0 + R_p \, U_p \end{cases}$$

La soluzione di questo problema può essere ottenuta analiticamente (utilizzando ad esempio la tecnica dei moltiplicatori di Lagrange) e vale

$$\widehat{U}_p = R_p^{\dagger} \, (\xi_f - A_Z^p \, \xi_0) \ , \tag{9}$$

dove $R_p^{\dagger}$ è la matrice pseudo-inversa di R_p. In Matlab i comandi sono

```
Rp = Bd;
for i=2:p
    Rp = [Bd Ad*Rp];
end

u = pinv(Rp)*(xf - Ad^p*x0);
```

In questo modo abbiamo garantito il raggiungimento dello stato finale, ma non abbiamo garantito che questo stato venga mantenuto al passo successivo. Se vogliamo anche fare mantenimento si deve innanzitutto calcolare l'ingresso u_p (se esiste) che rende lo stato finale ξ_f d'interesse un equilibrio. Una volta assicurati che questo accada, si può esplicitare il vincolo di raggiungimento e mantenimento in $p+1$ passi dello stato desiderato come

$$\xi_{p+1} = A_Z \, \xi_p + B_Z u_p = A_Z \, (A_Z^p \, \xi_0 + R_p U_p) + B_Z u_p = \xi_f \, . \tag{10}$$

Con un procedimento del tutto analogo al precedente si può calcolare la soluzione analitica come

$$\widehat{U}_p = (A_Z \, R_p)^{\dagger} \left(\xi_f - B_Z u_p - A_Z^{p+1} \, \xi_0 \right) \, . \tag{11}$$

Il calcolo del vettore ottimo degli ingressi si ottiene in Matlab coi comandi

```
u = pinv(Ad*Rp)*(xf - Ad^(p+1)*x0 - Bd*up);
u = [up; u];
```

dove `up` è ottenuto nel modo visto in precedenza.

Costruendo il diagramma a blocchi di Figura 5 è possibile verificare l'effettivo raggiungimento e mantenimento della posizione desiderata del sistema, nonché l'errore fra il sistema continuo e quello discretizzato, che si azzera (come previsto) ai multipli del tempo di campionamento (si veda Figura 6 per il controllo e l'uscita ottenuta, e Figura 7 per l'errore fra sistema tempo-continuo ed uscita pianificata a tempo-discreto). **Nota**: nel blocco `From Workspace`, inserire il nome della variabile che si desidera utilizzare (che deve essere presente nello spazio di lavoro) durante la simulazione, ad esempio `uSim`; il formato può essere di diverso tipo (si consulti la guida del blocco stesso), in questo esempio si usa la versione matriciale in cui la prima colonna contiene un vettore di tempi e la seconda gli associati valori di controllo; il valore di `Sample Time` deve essere pari a `Ts`; si vuol mantenere il valore finale costante selezionando `Form output after final data value by: Holding final value`. La sequenza di comandi per generare le variabili di simulazione `uSim` e `ySim` può essere

```
% input for simulink
x0 = zeros(n,1);
timing = Ts*(0:size(u,1)-1).';
uSim = [timing, flipud(u)];
y_lin = lsim(sys,flipud(u),timing,x0,'zoh');
ySim = [timing, y_lin];
```

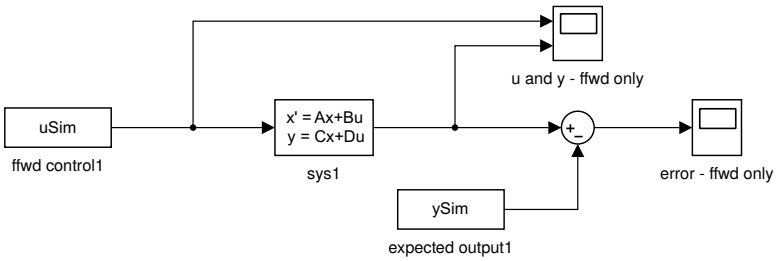

Figura 5: Schema Simulink del sistema lineare con ingresso ottimo.

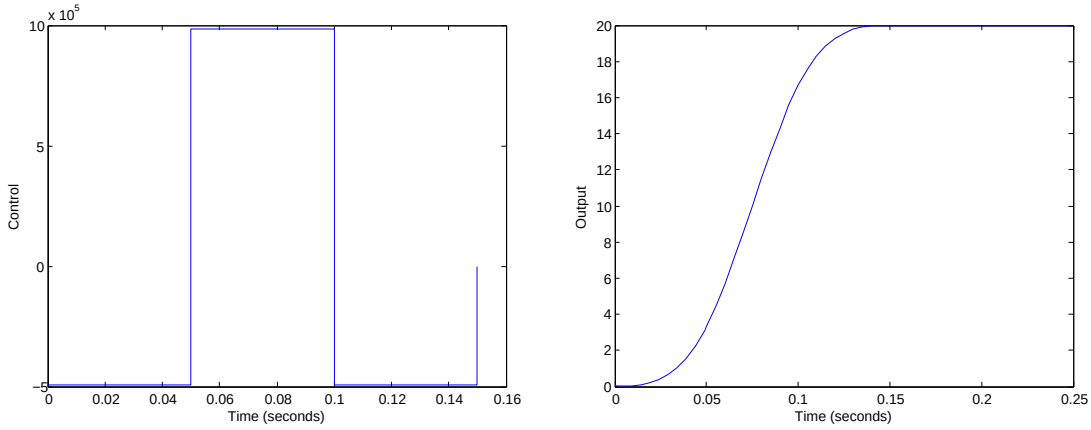

Figura 6: Sequenza ottima di controllo (a sinistra) ed uscita ottenuta in corrispondenza (a destra) ottenuta attraverso una simulazione con Simulink.

A.4.2 minima norma 2 - vincoli sul controllo $v \in [-20 \div 20]$ - tempo minimo $T_{\min,2}$

Quando si hanno vincoli sul massimo valore ammissibile del controllo, una possibilità consiste nell'aumentare l'orizzonte temporale (il numero di passi p) fin tanto che tali vincoli non sono soddisfatti.
Un semplice codice Matlab per fare ciò è il seguente

```
% bound on control variable
u_lb = -20; % lower bound
```

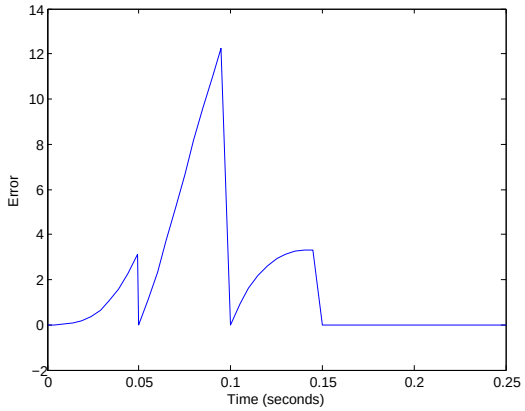

Figura 7: Errore fra uscita discreta ed uscita continua del sistema ottenuto attraverso una simulazione con Simulink. Si vede come tale errore sia nullo a valori multipli del tempo di campionamento.

```
u_ub = +20; % upper bound
% increase step number until the control is bounded
while not(and(min(u)>=u_lb,max(u)<=u_ub))
   p = p+1;
   Rp = [Bd Ad*Rp];
   u = pinv(Ad*Rp)*(xf - Ad^(p+1)*x0 - Bd*up);
end

disp(['Il tempo minimo ottenuto e'' pari a ' num2str(p*Ts)]);
```

da cui il tempo minimo ottenuto è pari a 5.65 sec ed il vettore di controlli associati, nonché la posizione ottenuta in simulazione, sono riportati in Figura 8.

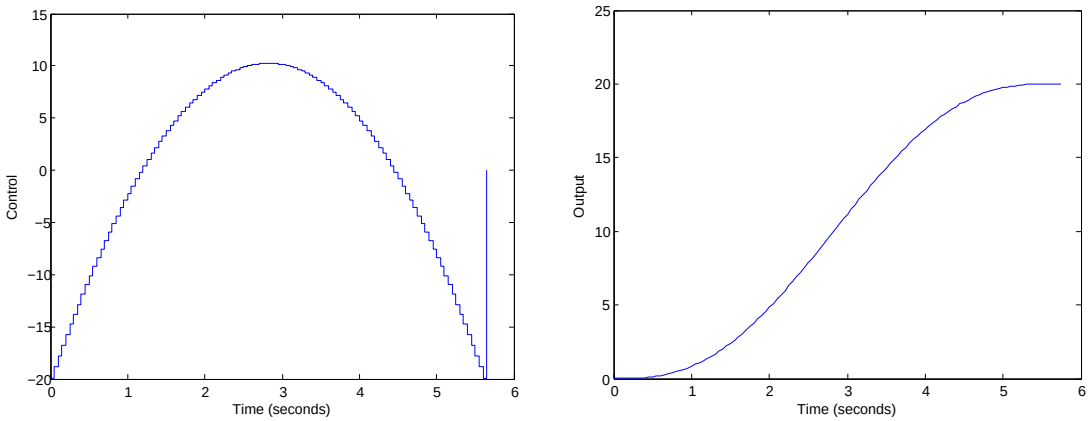

Figura 8: Vettore dei controlli (a sinistra) ed uscita associata (a destra) utilizzando il metodo della pseudo-inversa ma considerando un controllo limitato.

Un'altra possibilità consiste invece nel cambiare tecnica di ottimizzazione, utilizzando ottimizzazione ai minimi quadrati vincolati (con il comando Matlab `lsqlin`).
Poiché tale comando risolve un problema di ottimo del tipo

$$\begin{cases} \min\limits_{x} \|C\,x - d\|_2^2 \\ \text{soggetto a} \quad : \\ x_{\mathtt{lb}} \leq x \leq x_{\mathtt{ub}} \\ A_{\mathrm{eq}}\,x \leq b_{\mathrm{eq}} \\ A_{\mathrm{in}}\,x \leq b_{\mathrm{in}} \end{cases} \tag{12}$$

si deve porre la matrice C pari alla matrice identica ed il vettore d pari al vettore nullo per tornare a minimizzare la norma 2 del vettore di controllo.

Inoltre, per imporre il vincolo di raggiungimento dello stato finale si utilizzano la matrice A_{eq} ed il vettore b_{eq} come da (11), ovvero

$$A_{\text{eq}} = A_Z\, R_p \qquad b_{\text{eq}} = \xi_f - B_Z u_p - A_Z^{p+1}\, \xi_0\,. \tag{13}$$

Adesso basta imporre i vincoli sulla variabile di controllo attraverso i *lower bound* $x_{\texttt{lb}}$ e *upper bound* $x_{\texttt{ub}}$. Queste operazioni possono essere fatte in Matlab con il codice seguente

```
%% optimal control with constraints on u - lsqlin

% initialize flag to a non-acceptable value
exitflag = 0;
% cycle until constraints are satisfied
while(exitflag ~= 1)
    p = p + 1;
    Rp = [Bd Ad*Rp];

    Ccost = eye(p);
    d_cost = zeros(p,1);
    % to keep final state
    Aeq = Ad*Rp;
    beq = xf - Ad^(p+1)*x0 - Bd*up;

    % lower and upper bounds
    bound = 20;
    lb = -bound*ones(p,1);
    ub = bound*ones(p,1);

    % solution with lsqlin
    [u,~,~,exitflag] = lsqlin(Ccost,d_cost,[],[],Aeq,beq,lb,ub);
end
```

utilizzando inoltre un meccanismo del tutto analogo a quello visto in precedenza per la ricerca del tempo minimo, che risulta qui pari a 4.65 sec. Si veda la Figura 9 per il controllo ottenuto e l'uscita associata. Da notare come, avendo potuto porre un vincolo esplicito sui valori degli ingressi che si possono fornire al sistema, è stato possibile ottenere un tempo minimo inferiore rispetto al caso in cui questo vincolo non fosse esplicitato.

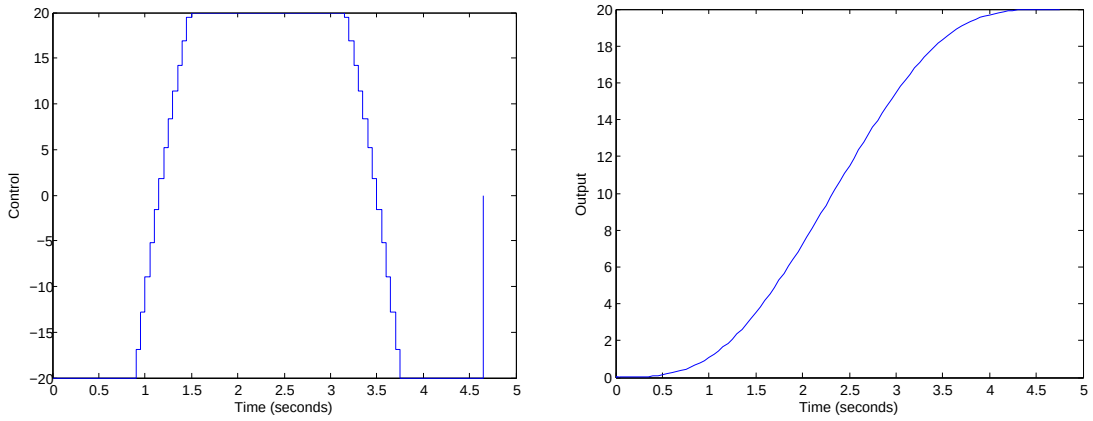

Figura 9: Vettore dei controlli ed uscita associata utilizzando il metodo dei minimi quadrati vincolati attraverso il comando `lsqlin`.

A.4.3 minima variazione massima dell'ingresso usando una funzione custom - tempo $T_{\min,2}+1$

Minimizzare la massima variazione dell'ingresso equivale a minimizzare la massima derivata dell'ingresso stesso, che può essere d'interesse in dipendenza dal tipo di attuatori utilizzati. Per fare ciò, il costo da minimizzare è

$$\max_{i\in[0\div p]} \|u_{i-1} - u_i\| \tag{14}$$

ovvero la norma infinito del vettore ottenuto come differenza di valori successivi dell'ingresso, considerando che prima e dopo il tempo di ottimizzazione l'ingresso assume valore zero ($u_{-1} = u_p = 0$). Questo può essere fatto in Matlab definendo una opportuna funzione

```
function out = myFun(in)
   % add zeros before initial and after final value
   u = [0; in; 0];
   % take the maximum variation
   out = norm(diff(u),inf);
end
```

che verrà poi minimizzata attraverso il comando `fmincon`, nel modo seguente

```
% double number of steps w.r.t. previous value
p = 93+1/Ts; % 113 steps in total
% reachability matrix in p steps
Rp = Bd;
for i=2:p
    Rp = [Bd Ad*Rp];
end

% to keep final state
Aeq = Ad*Rp;
beq = xf - Ad^(p+1)*x0 - Bd*up;

% lower and upper bounds
bound = 20;
lb = -bound*ones(p,1);
ub = bound*ones(p,1);

% function handle to use in the optimization
fOpt_handle = @(U)(myFun(U));

% initial guess for the control vector
u0 = zeros(p,1);

% optimization options
myoptions = optimset('Algorithm','Interior-point',...
                     'MaxFunEvals',80000);

u = fmincon(fOpt_handle,u0,[],[],Aeq,beq,lb,ub,[],...
            myoptions);
```

Da notare che il comando `fmincon` (di cui si suggerisce guardare approfonditamente l'help in Matlab e gli esempi correlati all'utilizzo per comprendere al meglio come creare handle di funzioni anche nel caso abbiano più parametri) può usare anche ulteriori vincoli di uguaglianza e disuguaglianza non lineari, non considerati in questo esempio.
In Figura 10 si vedono i controlli per uno stesso numero di passi $p = 93 + 1/Ts = 113$ (pari al tempo minimo trovato al punto precedente più un secondo) ottenuti sia con `lsqlin` che con `fmincon`, in modo da mettere in risalto come il cambio di indice di costo incida sulla soluzione del problema con stesso vincolo di raggiungimento dello stato finale (eccetto nel caso di tempo minimo, dove soltanto i vincoli hanno influenza). Nello specifico, minimizzare la massima variazione dell'ingresso porta ad ottenere un profilo più *morbido*.

A.4.4 aggiunta di vincoli sullo stato

Per poter aggiungere, in una minimizzazione rispetto alla variabile di controllo, un vincolo (o termine di costo, equivalentemente) dipendente dagli stati, è necessario esplicitare gli stati di interesse in funzione del vettore degli ingressi.
Presa opportunamente una matrice di uscita che ci dia la combinazione desiderata degli stati, ad esempio $C_v = [0\ 1\ 0]$ per ottenere la velocità di discesa ξ_2, si ha che il vettore V_Z che ne impila la sequenza ad ogni

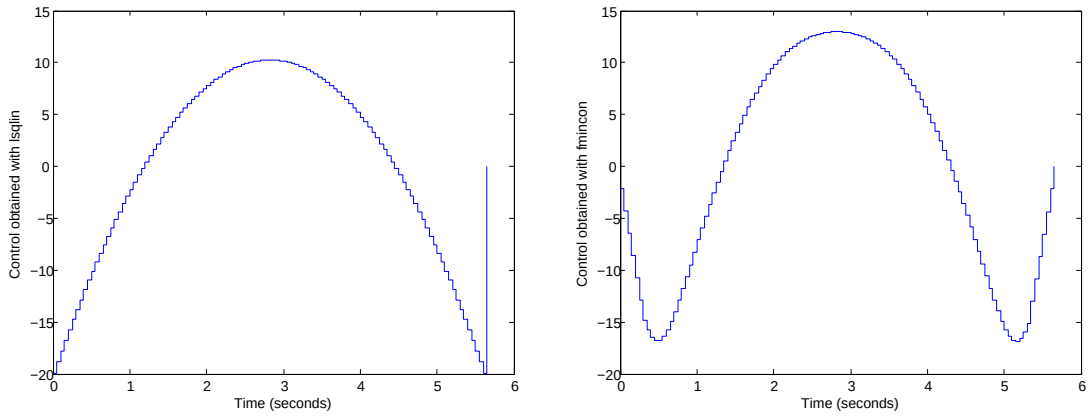

Figura 10: Vettore dei controlli ottenuto attraverso il comando `lsqlin` (a sinistra) e `fmincon` (a destra) per uno stesso orizzonte temporale. Si può vedere l'effetto di cambiare l'indice di costo sul profilo di controllo ottenuto.

istante (in ordine inverso) si può esprimere in funzione dei soli stato iniziale e ingressi come

$$V_Z = \begin{bmatrix} \xi_{2,p} \\ \vdots \\ \xi_{2,1} \end{bmatrix} = \begin{bmatrix} C_v \xi_p \\ \vdots \\ C_v \xi_1 \end{bmatrix} = \mathcal{H}_p U_p + \mathcal{O}_p \xi_0 \tag{15}$$

dove

$$\mathcal{H}_p = \begin{bmatrix} C_v B_Z & C_v A_Z B_Z & C_v A_Z^2 B_Z & \cdots & C_v A_Z^{p-1} B_Z \\ 0 & C_v B_Z & C_v A_Z B_Z & \cdots & C_v A_Z^{p-2} B_Z \\ 0 & 0 & C_v B_Z & \cdots & C_v A_Z^{p-3} B_Z \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & C_v B_Z \end{bmatrix}, \quad \mathcal{O}_p = \begin{bmatrix} C_v A_Z^p \\ C_v A_Z^{p-1} \\ C_v A_Z^{p-2} \\ \vdots \\ C_v A_Z \end{bmatrix} \tag{16}$$

e ξ_0 è lo stato iniziale, pari nel caso specifico di interesse a $[0\ 0\ 0]^{\mathrm{T}}$.
Le matrici $\mathcal{H}_p$ e $\mathcal{O}_p$ possono essere ottenute in Matlab con le seguenti istruzioni

```
%% Compute Hp and Op matrixes

% output matrix
Cv = [0 1 0];

% initialize matrixes
Hp_row = Cv*Rp; % last inserted row in Hp matrix
Hp = Hp_row;
Op = Cv*Ad;
% depend on input-output dimensions
my = size(Cv,1);
mu = size(Bd,2);

% build matrixes
for i = 2:steps
    Op = [Op*Ad; Cv*Ad];
    Hp_row = [zeros(my,mu) Hp_row(:,1:end-mu)];
    Hp = [Hp; Hp_row]; %#ok
end
```

Una volta ottenuto il vettore degli stati V_Z in funzione del vettore degli ingressi U_p, è possibile imporre il limite sugli stati nel modo seguente

$$v_{\mathtt{lb}} \leq V_Z \leq v_{\mathtt{ub}} \quad \Longrightarrow \quad v_{\mathtt{lb}} \leq \mathcal{H}_p U_p + \mathcal{O}_p \xi_0 \leq v_{\mathtt{ub}}\,, \tag{17}$$

dove $v_{\mathtt{lb}}$ e $v_{\mathtt{ub}}$ sono limiti vettoriali di dimensione opportuna; è inoltre possibile trasformare il tutto in un'unica disuguaglianza sul vettore di controllo come

$$\begin{bmatrix} \mathcal{H}_p \\ -\mathcal{H}_p \end{bmatrix} U_p \leq \begin{bmatrix} v_{\mathtt{ub}} - \mathcal{O}_p \xi_0 \\ -v_{\mathtt{lb}} + \mathcal{O}_p \xi_0 \end{bmatrix}, \tag{18}$$

disuguaglianza che può essere gestita come vincolo dalle routine di ottimizzazione viste in precedenza. Il codice Matlab già visto va dunque opportunamente integrato con

```
%% Linera inequality constraints

% bound on velocity
v_lb = -6*ones(steps,1);
v_ub = 6*ones(steps,1);

% inequality matrix and vector
Aineq = [Hp; -Hp];
bineq = [v_ub - Op*x0; -v_lb + Op*x0];

% do the optimization
u = fmincon(fOpt_handle,u0,Aineq,bineq,Aeq,beq,lb,ub,[],...
            myoptions);
```

Utilizzando poi le stesse istruzioni già viste per effettuare una simulazione lineare (comando `lsim`) all'interno di Matlab, si può controllare come sia variato il profilo di velocità ottenuto attraverso l'ottimizzazione: in Figura 11 si vedono i profili di velocità nel caso di ottimizzazione con e senza vincoli sullo stato. Il codice per ottenere tali figure, da usare dopo ognuna delle due ottimizzazioni ed associata simulazione lineare, è

```
figure
plot((0:p-1).'*Ts, diff(y_lin)./Ts)
xlabel('Time (seconds)');
ylabel('Velocity');
```

con cui si utilizza l'approssimazione della derivata con differenze finite.

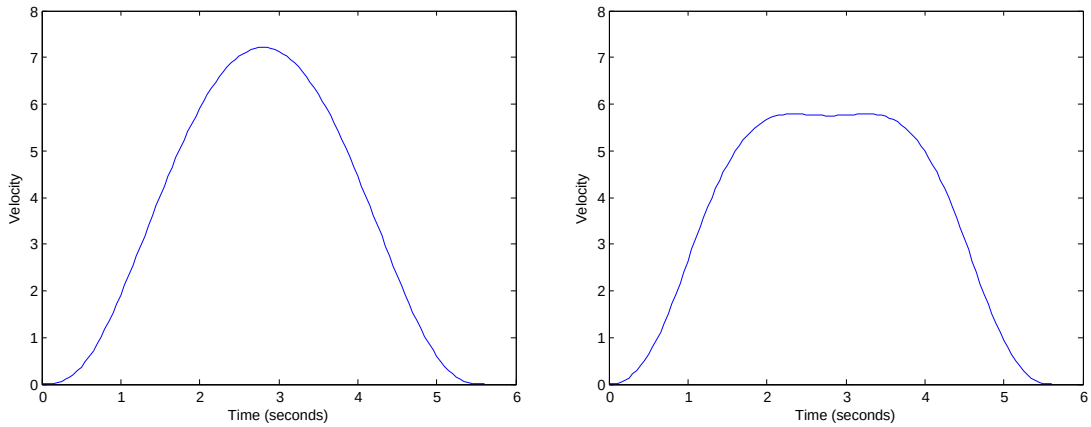

Figura 11: Profili di velocità ottenuti attraverso il comando `fmincon` senza (a sinistra) e con (a destra) vincoli di disuaguaglianza lineari sullo stato. Si può vedere come il vincolo richiesto venga rispettato dalla seconda ottimizzazione.

A.5 Simulazione del sistema con ingresso ottimo in feed-forward e controllore in feedback

Per poter utilizzare contemporaneamente il controllo in feedback e la sequenza di ingressi pre-pianificata in feed-forward è necessario utilizzare un opportuno montaggio per la simulazione. Nello specifico, oltre alle varie considerazioni già fatte su particolari blocchi Simulink che rimangono ancora valide, è necessario ricordare che l'ingresso al regolatore deve essere *l'errore rispetto alla traiettoria desiderata*, e non l'uscita, in quanto se l'uscita seguisse fedelmente la traiettoria desiderata senza bisogno di controllo, il regolatore non dovrebbe effettuare nessuna azione.

A.5.1 Caso lineare

Il montaggio più semplice nel caso lineare si può ottenere come in Figura 12, dove si nota l'assenza di un riferimento come nel caso del solo feedback, e l'introduzione di una differenza con la traiettoria desiderata dell'uscita. Il controllo totale e l'uscita associata sono visibili in Figura 13: come si può

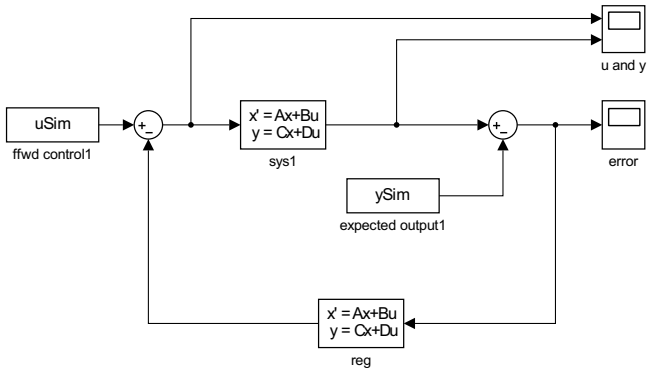

Figura 12: Modello Simulink per la simulazione lineare del sistema con ingresso ottimo in feed-forward e controllore in feedback.

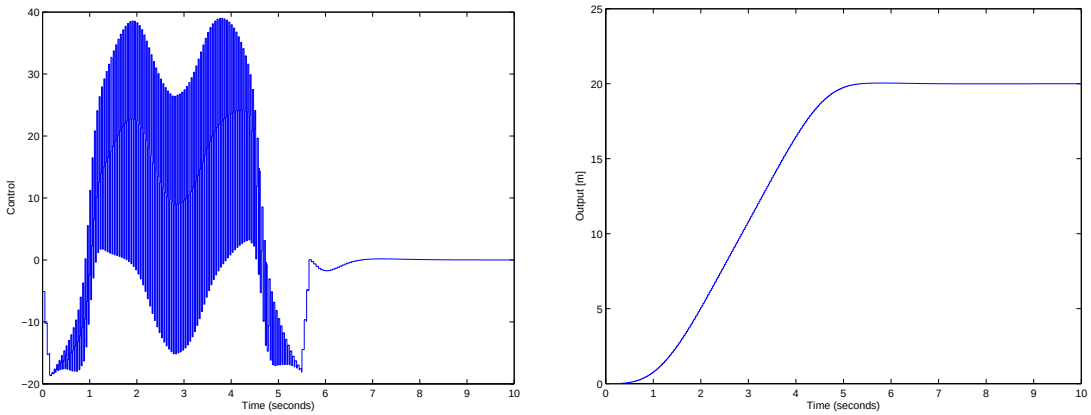

Figura 13: Controllo (a sinistra) e uscita associata (a destra) ottenuto con la simulazione lineare del sistema con ingresso ottimo in feed-forward e controllore in feedback.

osservare, il controllo totale non rispetta il limite imposto durante l'ottimizzazione. Per garantire che l'effettivo ingresso dato al sistema non superi il livello ammissibile, è necessario inserire una saturazione all'ingresso, come in Figura 14 che conterrà `bound` e `-bound` nei campi `Upper Limit` e `Lower Limit`, rispettivamente. Il controllo totale e l'uscita associata sono presentati in Figura 15.

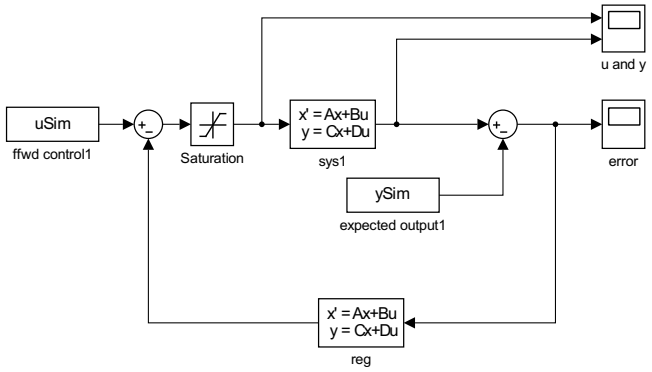

Figura 14: Modello Simulink per la simulazione lineare del sistema con ingresso ottimo in feed-forward e controllore in feedback, con l'aggiunta della saturazione sugli ingressi.

A.5.2 Caso non lineare

Il montaggio rispecchia quello del caso lineare, con alcune complicazioni dovute al fatto che le equazioni non lineari richiedono una diversa implementazione. Nello specifico è necessario creare una *Matlab function* che, dati come ingressi il controllo, il disturbo e gli stati del sistema restituisca il valore della derivata degli stati stessi, che dovrà poi essere integrato all'interno della simulazione. Una funzione di uscita (anch'essa possibilmente nonlineare $y = h(x)$, ma in questo caso si tratta soltanto di un selettore

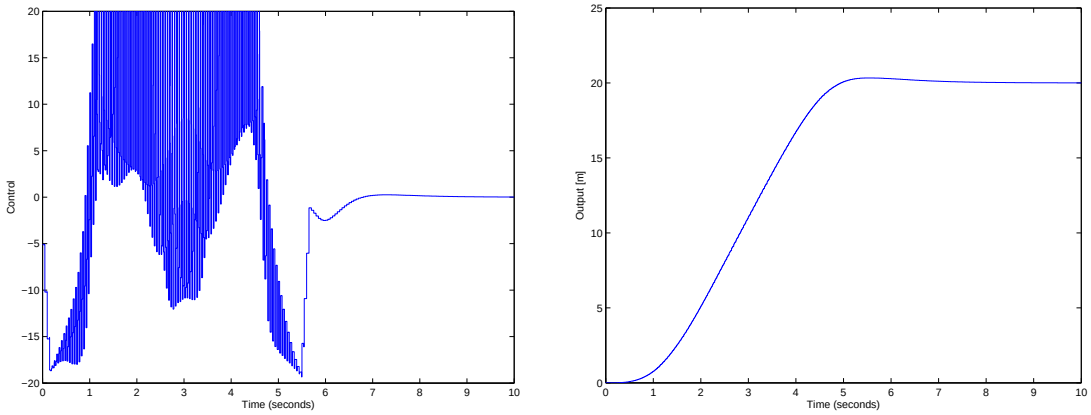

Figura 15: Controllo (a sinistra) e uscita associata (a destra) ottenuto con la simulazione lineare del sistema con ingresso ottimo in feed-forward e controllore in feedback con l'aggiunta della saturazione.

del primo elemento del vettore di stato) viene messa a valle dell'integratore per rispecchiare quella che è l'effettiva uscita del nostro sistema.
Inoltre, si deve decidere se implementare la dinamica non lineare originale, per cui lo stato iniziare dell'integratore sarà l'equilibrio trovato inizialmente al punto **A.1**, oppure la dinamica traslata, per cui lo stato iniziale dell'integratore sarà ξ_0.
Lasciando per esercizio il primo caso, la seconda implementazione può essere fatta come segue:

- uno script di inizializzazione di parametri geometrici e calcolo dell'equilibrio *init_system.m*

```
% init_system.m

% numerical parameters
m = 6e3;          % Kg
b = 0.03013;      % N s / m
alp = 0.981;    % m / s^2
bet = 1.329e5;  % s^-4
p0 = 1e5;         % Pa
rho = 1e3;        % Kg / m^3
g = 9.81;         % m / s^2
gam = 2.45e-6; % m s

% equilibria
z_bar = 300;     % m
F_bar = 0;       % N

ma_bar = alp/bet*(p0 + rho*g*z_bar);
u_bar = 0;
```

- una *Matlab function* contenente la dinamica del sommergibile *sommergibile.m*

```
function out = sommergibile_dyn(in)
% function out = sommergibile_dyn(in)
%
% Implement nonlinear system dynamics translated at the equilibrium

% Assign input vector to different variables
u1 = in(1);
u2 = in(2);
x1 = in(3);
x2 = in(4);
x3 = in(5);

% numerical parameters
init_system;
```

```
% equations
Dx1 = x2;
Dx2 = -b/m*x2 + alp - bet*(x3+ma_bar)/( p0 + rho*g*(x1+z_bar) ) ...
        -(u2 + F_bar)/m;
Dx3 = (p0 + rho*g*(x1+z_bar))*gam*(u1+u_bar);

out = [Dx1; Dx2; Dx3];
```

Il montaggio si può ottenere come in Figura 16, che differisce rispetto al caso lineare solo per la sostituzione del blocco *state-space* che implementava il sistema lineare con un blocco *Interpreted Matlab function*, un integratore ed un selettore per implementare la dinamica non lineare.
Il controllo totale e l'uscita associata sono presentati in Figura 17

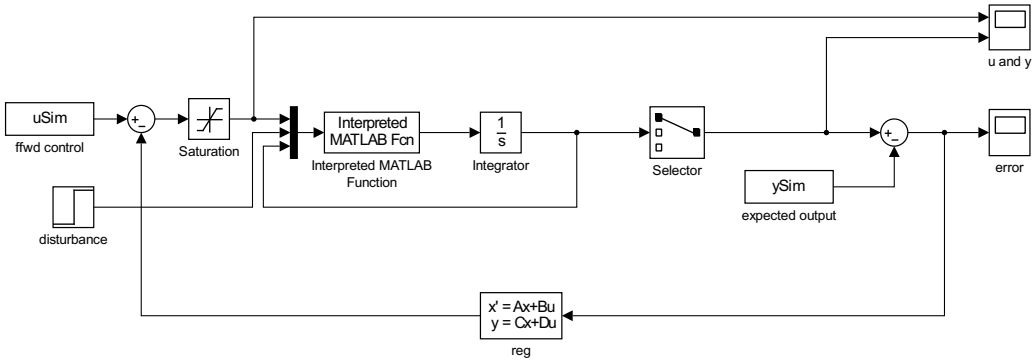

Figura 16: Modello Simulink per la simulazione non lineare del sistema con ingresso ottimo in feed-forward e controllore in feedback.

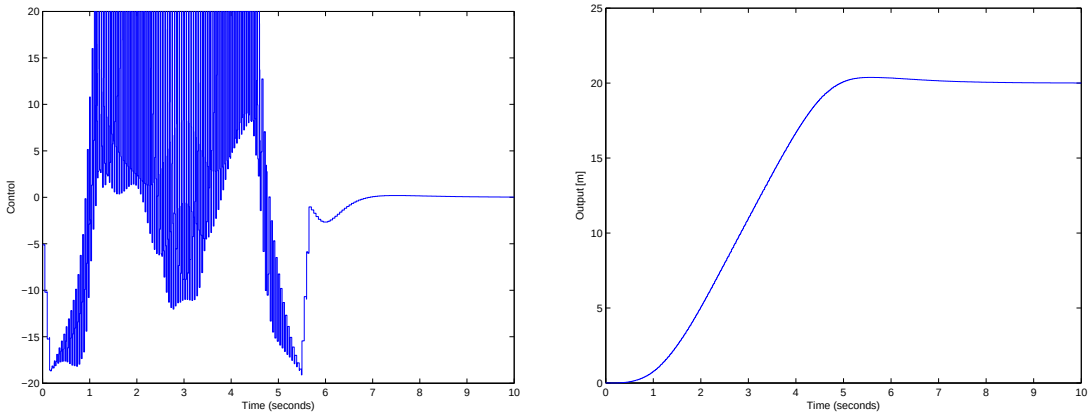

Figura 17: Controllo (a sinistra) e uscita associata (a destra) ottenuto con la simulazione non lineare del sistema con ingresso ottimo in feed-forward e controllore in feedback.