

# Dirigibile Termico

Andrea Marino

Settembre, 2020



Figura 1: Skyatch Personal Blimp

## Descrizione del sistema

Un dirigibile è un mezzo di trasporto che sfrutta la spinta di Archimede per galleggiare nell'aria e restare in quota. Questa spinta, nel dirigibile termico, nasce dalla diversa densità tra aria interna ed esterna ottenuta riscaldando l'ambiente chiuso con dei bruciatori.

L'ambiente esterno viene modellato secondo le indicazioni dell'*International Standard Atmosphere (ISA)* [3], le quali suggeriscono un andamento di pressione, temperatura e densità dell'aria dipendenti dall'altitudine. Vengono riportate le

formule simboliche i cui valori delle varie grandezze seguiranno nell'apposito paragrafo.

$$\begin{cases} T(z) = T_0 - L_o \cdot z \\ p(z) = p_0 \cdot (1 - \frac{L_o}{T_0} \cdot z)^{\frac{g}{R \cdot L_o}} \\ \rho(z) = \frac{p(z)}{R \cdot T(z)} \end{cases} \quad (1)$$

La massa complessiva di questo sistema è data dalla somma di un termine costante, che rappresenta il peso da sgonfio (struttura metallica, tessuto, cabina pilota e passeggero, motori e vano motori), e di una parte che dipende dalla densità dell'aria interna e, quindi, dalla quota a cui si trova il dirigibile.

$$m = m_0 + \rho_i V = m_0 + \rho \frac{T}{T_i} V$$

È possibile descrivere questo sistema fisico con due equazioni differenziali

$$\begin{cases} c\dot{T}_i = v - k_T(T_i - T) \\ m\ddot{z} = \rho V g + \rho T V \frac{\dot{T}_i}{T_i^2} \dot{z} - mg - \frac{1}{2} C_z S_z \rho \dot{z}^2 - F_z \end{cases} \quad (2)$$

Si nota che

- La prima equazione è un bilancio energetico dove  $v$  è la potenza termica che viene fornita dal bruciatore,  $c\dot{T}_i$  è la variazione di temperatura dell'aria interna e  $k_T(T_i - T)$  un termine di perdita proporzionale al  $\Delta T$  causato dal non perfetto isolamento della parete di tessuto;
- La seconda equazione viene ottenuta dalla formula generale del secondo principio di Newton  $\dot{Q} = \sum F$ :  $\rho V g$  è la spinta di Archimede,  $\rho T V \frac{\dot{T}_i}{T_i^2} \dot{z}$  nasce dal fatto che la massa non è costante ed è positivo con un aumento di temperatura,  $mg$  è il peso,  $\frac{1}{2} C_z S_z \rho \dot{z}^2$  è l'attrito aerodinamico e  $F_z$  è un eventuale disturbo che agisce verticalmente sul sistema.

Alcuni dei parametri sono stati presi dal data sheet [4] di un dirigibile termico reale, il *Skyatch Personal Blimp* (fig. 1), altri invece vengono dal modello ISA.

Per stimare il valore del coefficiente d'attrito viene maggiorato di poco quello della sfera (pari a 0.47).

Per stimare<sup>1</sup> il valore della conducibilità termica  $k_T$  si considera che il materiale è molto performante, associandogli una trasmittanza termica  $g$  pari a  $0.05 \text{ W/m}^2\text{K}$ : si ottiene che  $k_T = g \cdot S \simeq g \cdot (2\pi r^2 + 2\pi rl) \simeq 150 \text{ W/K}$ .

|    |             |                          |               |
|----|-------------|--------------------------|---------------|
| 1  | m0=2000;    | %kg                      | da data sheet |
| 2  | l=31.1;     | %m                       | da data sheet |
| 3  | r=11.15;    | %m                       | da data sheet |
| 4  | V=5.8e3;    | %m^3                     | da data sheet |
| 5  | T0=288.15;  | %K                       | da ISA        |
| 6  | L0=6.5e-3;  | %K/m                     | da ISA        |
| 7  | p0=1.013e5; | %Pa                      | da ISA        |
| 8  | g=9.81;     | %m/s^2                   |               |
| 9  | R=287.1;    | %m^2/(s^2*K)             |               |
| 10 | Cz=0.52;    | %per superficie laterale |               |
| 11 | Kt=150      | %W/K                     |               |
| 12 | c=718       | %J/K                     |               |

<sup>1</sup>Per semplificare il modello, il dirigibile viene considerato come se fosse un cilindro.

## Richieste

1. Determinare tutti gli equilibri fisicamente ammissibili per il sistema in assenza di disturbo
2. Supponendo di disporre della misura della quota  $z$ , determinare una realizzazione del sistema linearizzato attorno ad un equilibrio con il valore di  $\bar{z} = 500$  m
3. Progettare un compensatore basato sul regolatore che stabilizzi il sistema
4. Pianificare gli ingressi per portare il sistema in  $z = \bar{z} + 50$  e mantenerlo indefinitamente lì, discretizzando il sistema con un tempo di campionamento  $T_s = 0.05$  s. Effettuare ciò
  - (a) In assenza di vincoli sullo stato o sugli ingressi con un numero di passi (e quindi di tempo) definito
  - (b) Inserendo dei limiti fisici all'ingresso facendo in modo che  $-\bar{v} \leq v \leq 600$  W
  - (c) Minimizzando la massima variazione di ingressi tra due passi adiacenti aumentando di 5 s il tempo necessario
  - (d) Facendo appartenere la temperatura al range  $[-1.5 \div 1.5]^\circ\text{C}$ .
5. Simulare ogni tipo di pianificazione sul sistema linearizzato, sul sistema reale e sul sistema controllato
6. Stimare numericamente la RAS per il sistema controllato nella nuova posizione di equilibrio

## 1 Equilibri

Scegliendo come vettore dello stato  $x = [x_1 \ x_2 \ x_3]^T = [T_i \ z \ \dot{z}]^T$  e come vettore degli ingressi  $u = [u_1 \ u_2]^T = [v \ F_d]^T$  si possono riscrivere le equazioni in forma normale

$$\begin{cases} \dot{x}_1 = \frac{u_1}{c} - \frac{k_T}{c}(x_1 - T) \\ \dot{x}_2 = x_3 \\ \dot{x}_3 = \frac{\rho V g + \rho T V \frac{x_1}{x_2} \dot{x}_3 - mg - \frac{1}{2} C_z S_z \rho x_3^2 - u_2}{m} \end{cases} \quad (3)$$

Per trovare le posizioni di equilibrio in assenza di disturbo si risolve il sistema  $\dot{x} = 0$  con  $u_2 = 0$ ; così facendo si ottiene

$$\begin{cases} \bar{u}_1 = \bar{v} = k_T(\bar{x}_1 - \bar{T}) \\ \bar{x}_3 = \bar{z} = 0 \\ \bar{x}_1 = \bar{T}_i = \frac{\bar{T}}{1 - \frac{m_0}{\rho V}} \end{cases} \quad (4)$$

In questo modo si ha

$$\bar{x} = \begin{bmatrix} 404.4 \\ 500 \\ 0 \end{bmatrix} \quad \bar{u} = \begin{bmatrix} 17.9 \cdot 10^3 \\ 0 \end{bmatrix}$$

con la temperatura espressa in K e l'ingresso in W.

## 2 Linearizzazione attorno alla posizione di equilibrio

Una realizzazione in forma di stato è riportata nella formula 3; per traslarla attorno alla posizione di equilibrio le attuali variabili di stato vengono sostituite da  $\tilde{x} = [\tilde{x}_1 \ \tilde{x}_2 \ \tilde{x}_3]^T = [x_1 - \bar{x}_1 \ x_2 - \bar{x}_2 \ x_3]^T$  e il vettore degli ingressi da  $\tilde{u} = [\tilde{u}_1 \ \tilde{u}_2]^T = [u_1 - \bar{u}_1 \ 0]^T$  in modo da ottenere

$$\begin{cases} \dot{\tilde{x}}_1 = \frac{\tilde{u}_1}{c} - \frac{k_T}{c} \tilde{x}_1 \\ \dot{\tilde{x}}_2 = \tilde{x}_3 \\ \dot{\tilde{x}}_3 = \frac{\rho V g + \rho T V \frac{\dot{\tilde{x}}_1}{(\bar{x}_1 + \tilde{x}_1)^2} \dot{\tilde{x}}_3 - mg - \frac{1}{2} C_z S_z \rho x_3^2}{m} \end{cases} \quad (5)$$

In queste nuove coordinate si ha che l'origine è il nuovo punto di equilibrio.

Il sistema viene approssimato con un sistema lineare scritto nella forma di stato come

$$\begin{aligned} \dot{x} &= f(x) \simeq Ax + Bu \\ y &= h(x) \simeq Cx + Du \end{aligned} \quad (6)$$

con

$$A = \left. \frac{\partial f}{\partial x} \right|_{\substack{x=\bar{x} \\ u=\bar{u}}} \quad B = \left. \frac{\partial f}{\partial u} \right|_{\substack{x=\bar{x} \\ u=\bar{u}}} \quad C = \left. \frac{\partial h}{\partial x} \right|_{\substack{x=\bar{x} \\ u=\bar{u}}} \quad D = \left. \frac{\partial h}{\partial u} \right|_{\substack{x=\bar{x} \\ u=\bar{u}}} \quad (7)$$

da cui, ricordando che si ha misura della sola altezza  $z$  (per cui  $y = x_2$ ), si ottiene la rappresentazione in forma numerica delle matrici

$$A = \begin{bmatrix} -0.209 & -1.4 \cdot 10^{-3} & 0 \\ 0 & 0 & 1 \\ 0.017 & -1.24 \cdot 10^{-4} & 1.05 \cdot 10^{-17} \end{bmatrix} \quad B = \begin{bmatrix} 1.4 \cdot 10^{-3} & 0 \\ 0 & 0 \\ 0 & 1.48 \cdot 10^{-4} \end{bmatrix}$$

$$C = [0 \ 1 \ 0] \quad D = [0]$$

La matrice  $A$  ha autovalori  $\lambda_i = [-0.209, 2.6 \cdot 10^{-4} \pm i1.5 \cdot 10^{-2}]^T$  per cui il sistema linearizzato non è stabile. Di conseguenza, per il teorema indiretto di Lyapunov, non lo è neanche il sistema originale. Questa instabilità è dovuta al fatto che, ad esempio, se il dirigibile arriva nella posizione di equilibrio con velocità diversa da zero continuerà a salire: in questo modo l'ingresso non è più in grado di garantire l'equilibrio (dovendo compensare perdite maggiori); ciò causa una diminuzione della spinta di Archimede che comporta un'accelerazione verso il basso con la conseguente caduta.

Diventa necessario quindi inserire un controllore che stabilizzi il sistema attorno alla posizione di equilibrio.

## 3 Controllore stabilizzante

Per stabilizzare il sistema viene utilizzato un compensatore basato su regolatore.

Come prima cosa bisogna verificare che il sistema lineare sia completamente osservabile e completamente raggiungibile con il seguente script dove, eseguiti i calcoli, si nota che effettivamente le matrici hanno rango pieno.

```

1 sys=ss(A,B,C,D);
2
3 %Raggiungibilità
4 rR=rank(ctrb(A,B));
5 disp(['rR=' num2str(rR)]);
6
7 %Osservabilità
8 rO=rank(obsv(A,C));
9 disp(['rO=' num2str(rO)]);

```

Nella progettazione del compensatore bisogna tenere conto che

- È richiesta una certa prontezza di risposta del sistema, perché potrebbero insorgere dei problemi per cui è necessaria una rapida variazione di quota: questo può essere tradotto in un tempo di assestamento al 5% al gradino in 3 s
- Oscillazioni attorno alla posizione di equilibrio potrebbero essere sgradevoli per il pilota: questo può essere realizzato con un controllore ad un polo dominante
- Poli per la convergenza della stima ( $A+LC$ ) molto più veloci di quelli della stabilità ( $A+BK$ )

Come conseguenza di queste specifiche, il valore dei poli che influenzano la stabilità dovrà essere tale per cui

$$\omega_i \geq \frac{3}{T_{a,5}} = 1 \quad \wedge \quad \omega_1 \ll \omega_2, \omega_3 \quad (8)$$

per cui una possibile scelta può essere  $p = [-1.5 \ -10 \ -12]^T$ . I poli per la convergenza della stima vengono scelti come  $q = 5 \cdot p$ . In Matlab questo viene eseguito con queste istruzioni

```

1 % Poli di A+BK ... stabilità
2 p=[-1.5;-10;-12];
3 K=-place(A,B,p);
4
5 % Poli d A+LC ... convergenza
6 q=5*p;
7 L=- (place(A.',C.',q)).';
8
9 %Regolatore parallelo
10 reg=ss(A+B*K+L*C,-L,-K,0);

```

dove è stata adottata la struttura per il montaggio del regolatore in parallelo. Per controllare che le specifiche siano rispettate è possibile testare il sistema retroazionato con un ingresso a gradino

```

1 %% Anello chiuso e valutazione risposta al gradino
2 Gc1=feedback(sys,reg);
3 opt=stepDataOptions('StepAmplitude',dcgain(reg));
4 step(Gc1,opt)

```

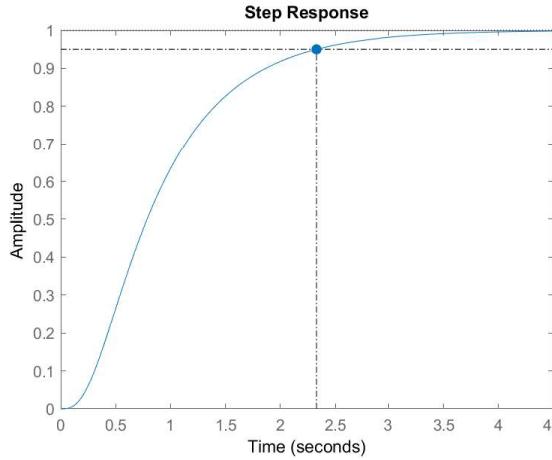


Figura 2: Step response con banda del 5% e tempo di assestamento.

## 4 Discretizzazione, pianificazione ottima e simulazione

### 4.1 Discretizzazione

Per poter valutare la sequenza di ingressi da dare al sistema è necessario discretizzarlo. Essendo un sistema LTITC soggetto ad ingressi costanti a tratti si può procedere con la tecnica dello *Zero Order Hold* che produce valori esatti ad ogni multiplo del tempo di campionamento.

$$\begin{aligned} x((k+1)T_s) &= e^{T_s A} x(kT_s) + \left( \int_0^{T_s} e^{A(T_s-\tau)} d\tau \right) Bu(k) \\ x_Z(k+1) &= A_z x_z(k) + B_z u(k) \end{aligned} \quad (9)$$

La discretizzazione può alterare alcune proprietà strutturali del sistema come la stabilità (instabilità in questo caso) o la raggiungibilità. Gli autovalori della matrice  $A_d$  sono  $\lambda_{i_d} = [0.99 \ 1 \pm 8 \cdot 10^{-4}]^T$  quindi l'instabilità non viene alterata; per controllare che sia anche quello discretizzato raggiungibile si valuta la raggiungibilità con le nuove matrici  $A_d$  e  $B_d$  e risulta anch'esso raggiungibile.

```

1 %% Discretizzazione
2 Ts=0.05;
3 sysD=c2d(sys,Ts,'zoh');
4 Ad=sysD.a;
5 Bd=sysD.b;
6
7 %Raggiungibilita'
8 Rd=ctrb(Ad,Bd);
9 if rank(Rd)<3
10     warning('Il sistema discretizzato non e\' raggiungibile')
11 else
12     disp('Il sistema discretizzato e\' raggiungibile')
13 end

```

## 4.2 Simulazioni

La dinamica del sistema viene simulata con il modello Matlab Simulink riportato in figura 3. Si nota un blocchetto di richiamo della variabile *uSim* dal workspace, creato negli script, che contiene le coppie tempo - ingresso, collegato a due sottosistemi e ad uno *scope* utilizzato per una visualizzazione gli ingressi. I due sottosistemi sono

- Sistema senza regolatore, figura 4, nel quale il vettore *uSim* viene dato come input ai vari sistemi non stabilizzati (non lineare, lineare e lineare discretizzato). Si può notare che prima dell'ingresso dell'*Interpreted Matlab Function* associata al sistema non lineare viene inserito un blocco di saturazione per la velocità limite che può raggiungere il dirigibile, pari a

$$V_l = \sqrt{\frac{2mg}{\rho A C_z}} \simeq 17.8 \text{ m/s}$$

- Sistema con regolatore, figura 5, nel quale l'ingresso che viene fornito al sistema non lineare viene ottenuto dalla differenza tra ingresso calcolato ed uscita del regolatore, il quale cerca di annullare l'errore di linearizzazione: in questo modo il regolatore viene montato in parallelo. Per essere sicuri che l'ingresso effettivo sia nel range desiderato, viene inserito anche qui un blocco di saturazione

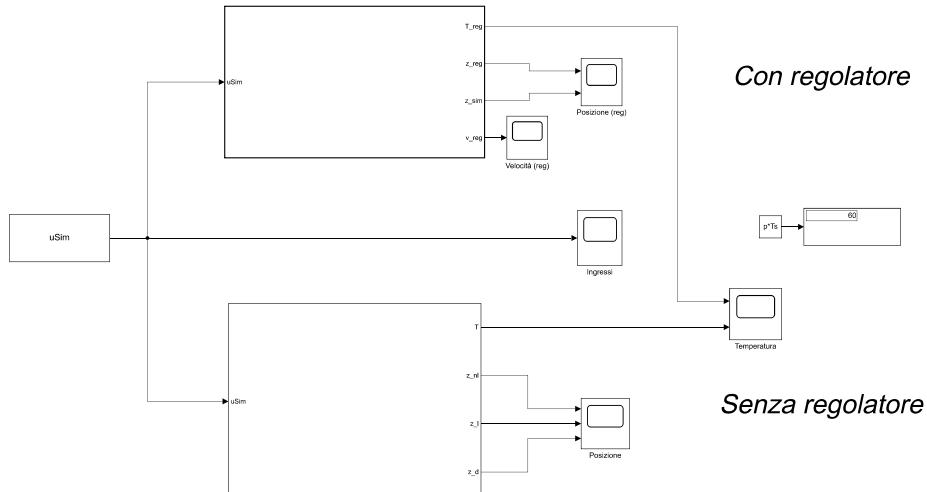


Figura 3: Modello complessivo per la simulazione

Vengono inseriti inoltre altri *scopes* con l'obiettivo di visualizzare i vari output delle simulazioni che verranno allegati al presente documento.

Al termine di ogni paragrafo vengono riportati i grafici di

- Serie di ingressi calcolati
- Andamento della *z* nel sistema non regolato (non lineare, linearizzato e discretizzato)
- Andamento della *z* nel sistema regolato

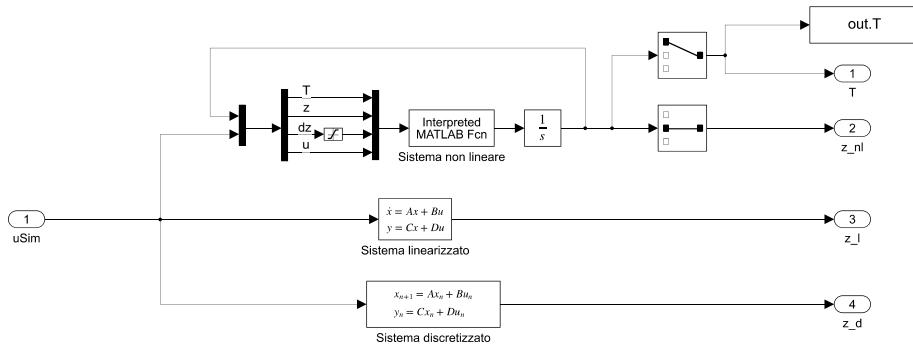


Figura 4: Sottosistema senza regolatore

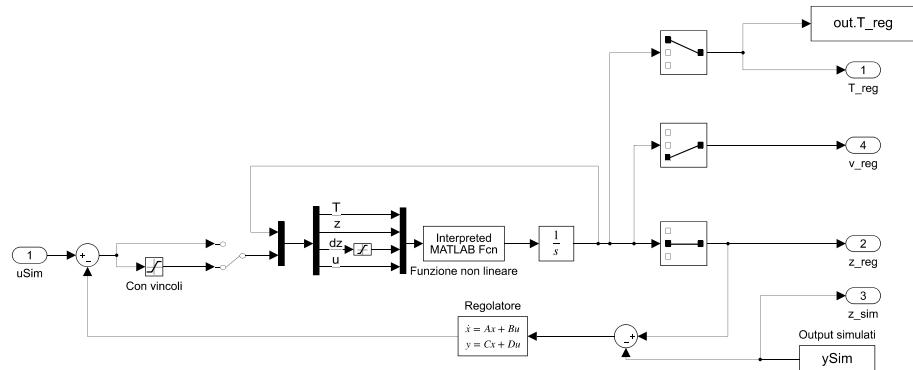


Figura 5: Sottosistema con regolatore

### 4.3 Pianificazione ottima non vincolata

L'obiettivo di questo e dei prossimi sottoparagrafi è calcolare una legge per l'ingresso che porti il sistema dalla posizione  $\bar{z} = 500$  m alla posizione  $z_f = \bar{z} + \Delta z = 550$  m, per poi mantenerla, rispettando determinati vincoli. In termini vettoriali, il sistema deve passare dallo stato iniziale  $x_0 = [0 \ 0 \ 0]^T$  allo stato  $x_f = [0 \ 50 \ 0]^T$ . Con il termine *ottima* si intende che verrà scelta tra tutte le possibili soluzioni quella che minimizza una stabilità funzione di costo.

Come primo approccio si sceglie di pianificare gli ingressi in modo tale da portare il sistema alla posizione finale e mantenerla in  $p = 1200$  passi, corrispondenti al tempo  $t = 60$  s, minimizzando la norma euclidea del vettore  $u$ . Questo può essere fatto considerando che

$$\begin{bmatrix} X(p-1) \\ X(p) \end{bmatrix} = \begin{bmatrix} A_d^{p-1}x_0 \\ A_d^p x_0 \end{bmatrix} + \begin{bmatrix} 0 & B_d & A_d B_d & \dots & A_d^{n-2} B_d \\ B_d & A_d B_d & A_d^2 B_d & \dots & A_d^{n-1} B_d \end{bmatrix} \begin{bmatrix} u(p) \\ u(p-1) \\ \vdots \\ u(0) \end{bmatrix} \quad (10)$$

Essendo  $x_0$  il vettore nullo, il termine  $\begin{bmatrix} A_d^{p-1}x_0 \\ A_d^p x_0 \end{bmatrix}$  sarà anch'esso nullo e quindi verrà sistematicamente omesso. Imponendo quindi l'uguaglianza dello stato ai passi  $p - 1$  e  $p$  e chiamando con  $\overline{R}_p^\dagger$  la pseudoinversa della matrice  $\begin{bmatrix} 0 & B_d & A_d B_d & \dots & A_d^{n-2} B_d \\ B_d & A_d B_d & A_d^2 B_d & \dots & A_d^{n-1} B_d \end{bmatrix}$ , il vettore degli ingressi diventa

$$\begin{bmatrix} u(p) \\ u(p-1) \\ \vdots \\ u(0) \end{bmatrix} = \overline{R}_p^\dagger \begin{bmatrix} x_f \\ x_f \end{bmatrix}.$$

In Matlab questo algoritmo può essere implementato con il codice riportato di seguito.

```

1 %% Matrice di raggiungibilità
2 for i=1:p
3     Rp_down(:,i)=Ad^(i-1)*Bd;
4 end
5 Rp=zeros(size(Bd)) Rp_down(:,1:end-1);Rp_down];
6
7 %% Pianificazione ottima
8 u=pinv(Rp)*([x_f;x_f]);
9 up=u(1);

```

In alternativa si sarebbe potuto ricavare il valore dell'ingresso di mantenimento  $u(p)$  direttamente dalla prima equazione del sistema 5 e risolvere la seconda riga dell'equazione 10.

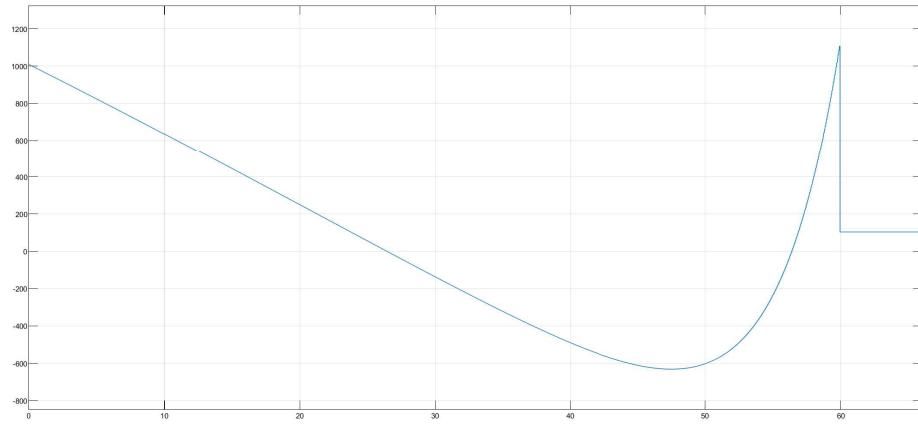


Figura 6: Ingressi in assenza di vincoli

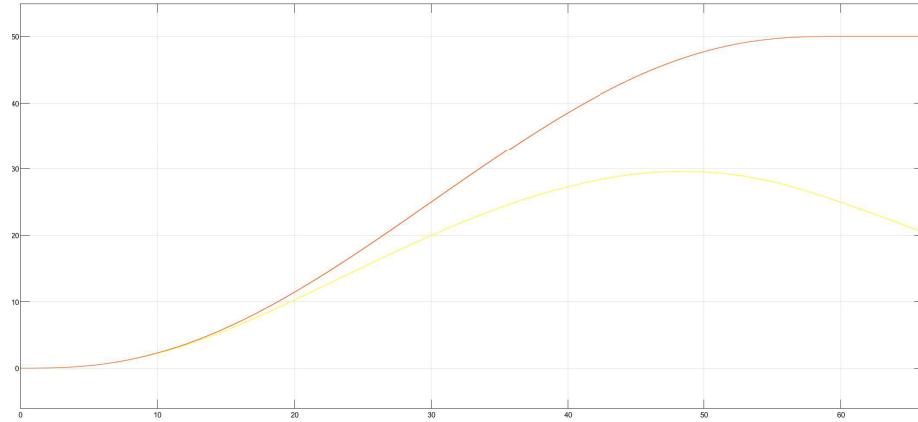


Figura 7: Andamento  $z$  nei tre sistemi senza regolatore (discreto e linearizzato sono indistinguibili qui)

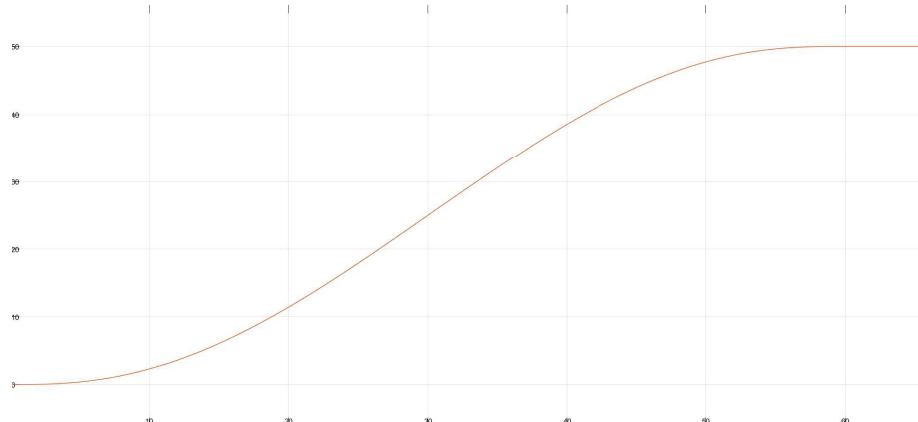


Figura 8: Andamento  $z$  nel sistema linearizzato e nel sistema non lineare con regolatore (sono indistinguibili qui)



Figura 9: Dettagli dei grafici

#### 4.4 Pianificazione ottima con vincoli sugli ingressi

Nel pianificare gli ingressi non si può non tenere conto di alcuni limiti fisici che gli stessi possono avere. In questo caso questi vincoli sono

- Il bruciatore non può sottrarre calore dall'aria, quindi, ricordando che tutta la pianificazione è stata effettuata attorno al punto di equilibrio a cui corrisponde un ingresso  $\bar{v}$ , occorre impostare che  $v \geq -\bar{v}$
- Il bruciatore non può fornire potenza illimitata, quindi c'è un limite superiore che viene fissato a 600 W per cercare di portare a circa la metà il massimo rispetto al punto precedente

Per realizzare questa pianificazione si può incrementare il numero di passi (e quindi il tempo) per raggiungere la posizione finale fin quando l'andamento degli ingressi non entra nella banda considerata. Dato che il valore di mantenimento è lo stesso per qualsiasi tipologia di pianificazione, è stato calcolato solo nella prima pianificazione e salvato come `up=u(1)`: così facendo si procede a risolvere solo la seconda equazione del sistema 10. Una possibile implementazione di questo algoritmo in Matlab è la seguente

```

1 %% Incremento del numero di passi
2 % Matrice di raggiungibilità
3 Rp=Bd;
4 for i=2:p
5     Rp=[Bd,Ad*Rp];
6 end
7
8 % Valori saturazione
9 u_min=-u_eq(1);           %lower bound
10 u_max=+600;              %upper bound
11
12 while not (and(min(u)>=u_min,max(u)<=u_max))
13     p=p+1;
14     Rp=[Bd Ad*Rp];
15     u=pinv(Ad*Rp)*(x_f-Bd*up);
16 end
17 u=[up;u];
18 disp(['Trovato un tempo minimo di ' num2str(p*Ts)]);

```

Questo modo di procedere porta ad incrementare la pianificazione a  $p = 1621$ , corrispondente a circa 81 secondi.

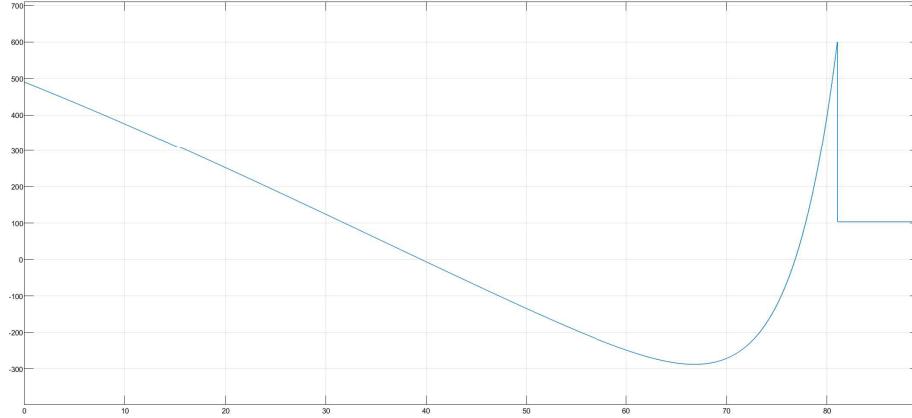


Figura 10: Ingressi con presenza di vincoli

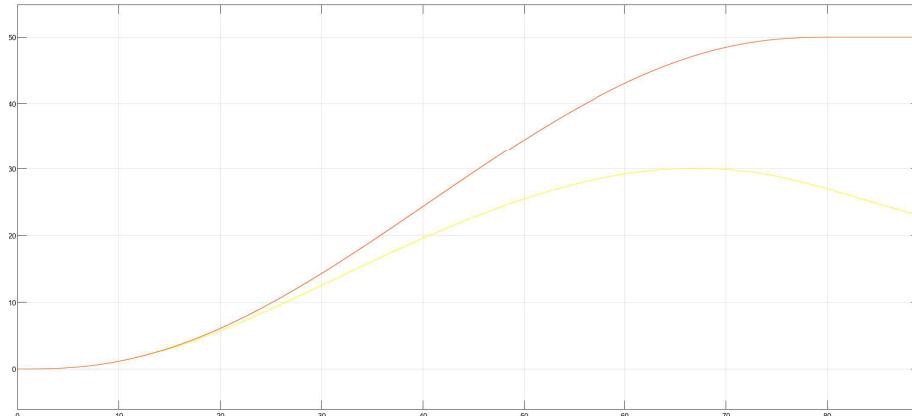


Figura 11: Andamento  $z$  nei tre sistemi senza regolatore

#### 4.5 Pianificazione ottima che minimizza la differenza tra due passi successivi

Per poter effettuare questa pianificazione occorre utilizzare una function di appoggio che prenda in ingresso un vettore e restituisca la norma infinita del vettore creato impilando la differenza tra un ingresso e il successivo. Questa funzione può essere implementata così

```

1 function out = FunOtt(in)
2 u=[0;in;0];
3 out=norm(diff(u),inf);
4 end

```

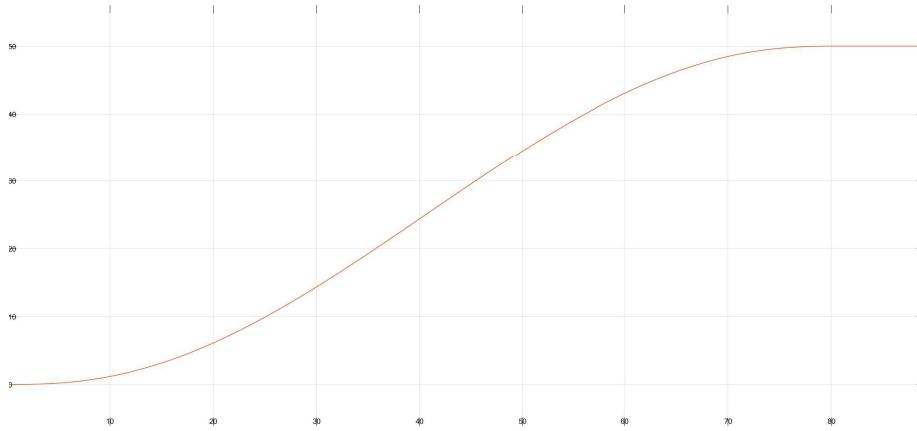


Figura 12: Andamento  $z$  nel sistema linearizzato e nel sistema non lineare con regolatore

A tale scopo viene utilizzata la routine `fmincon` che serve a minimizzare una funzione handle di input rispettando equazioni e disequazioni che rappresentano i vincoli. Un possibile modo per poter eseguire questa pianificazione è

```

1 % Aggiunta tempo
2 p=p+100;
3
4 % Raggiungibilita'
5 Rp=Bd;
6 for i=2:p
7     Rp=[Bd, Ad*Rp];
8 end
9 u=pinv(Ad*Rp)*(x_f-Bd*up);
10 Aeq=Ad*Rp;
11 beq=x_f-Bd*up;
12 lb=repmat(u_min,p,1);
13 ub=repmat(u_max,p,1);
14
15 % Tentativo iniziale di controllo
16 u0=zeros(size(u));
17
18 % Funzione handle da minimizzare
19 fun=@(u) (FunOtt(u));
20
21 %Parametri di ottimizzazione
22 options = optimset('Algorithm','interior-point','MaxFunEval',1e7);
23
24 u=fmincon(fun,u0,[],[],Aeq,beq,lb,ub,[],options);
25 u=[up;u];

```

A causa dell'elevato numero di passi, il tempo per il calcolo è molto elevato; viene pertanto inserito un limite di  $10^7$  tentativi, fiduciosi del fatto che il risultato sia buono e restituisca una funzione quanto più regolare possibile.

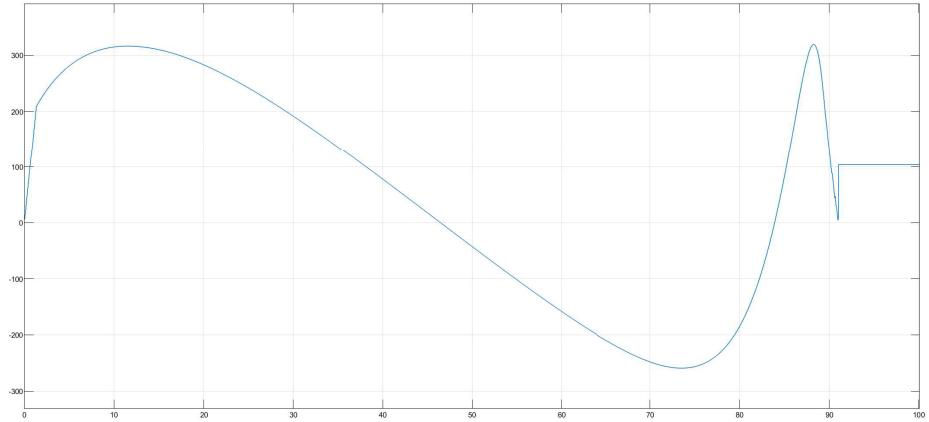


Figura 13: Ingressi con vincoli e ottimizzazione

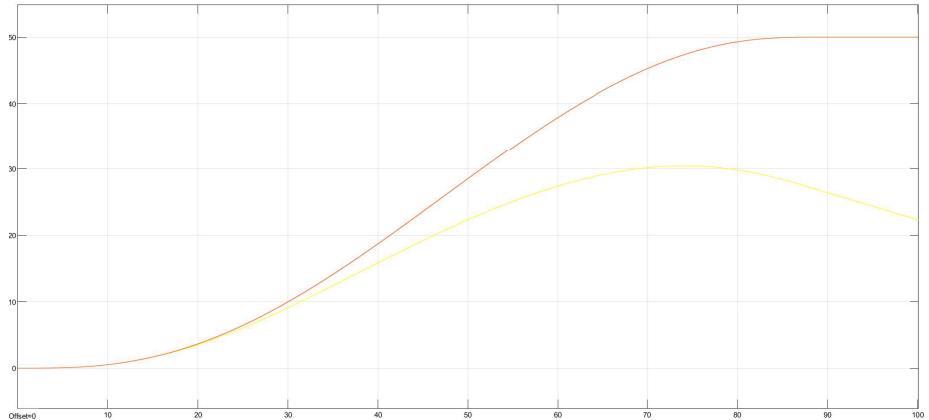


Figura 14: Andamento  $z$  nei tre sistemi senza regolatore

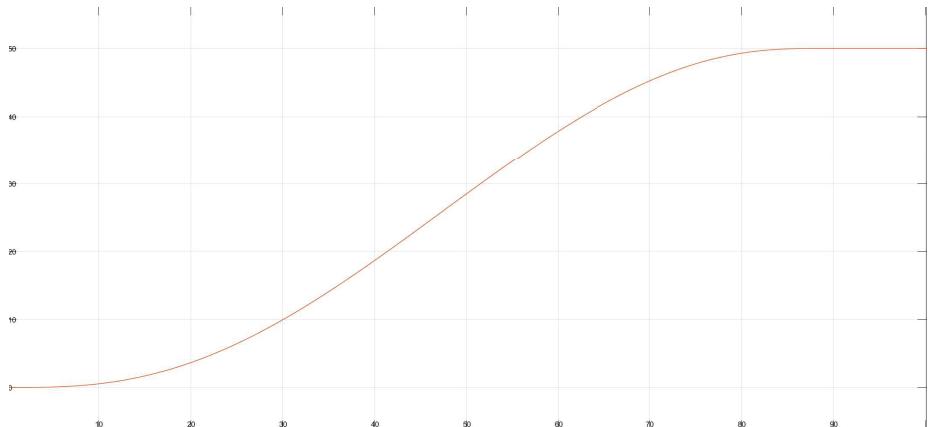


Figura 15: Andamento  $z$  nel sistema linearizzato e nel sistema non lineare con regolatore

## 4.6 Pianificazione ottima con vincolo sull'ingresso e sullo stato

L'ultima richiesta è quella di effettuare una pianificazione che limiti i valori assunti dello stato: per poter ridurre le perdite si stabilisce che la temperatura relativa non esca dall'intervallo  $[-1.5 \div 1.5]$  °C. A causa della non disponibilità della misura diretta della temperatura si può realizzare una pianificazione sul sistema lineare utilizzando la routine `fmincon` ovvero aggiungendo come ulteriore vincolo

$$\begin{bmatrix} H_p \\ -H_p \end{bmatrix} u \leq \begin{bmatrix} T_{max} - O_P x_0 \\ T_{min} - O_p x_0 \end{bmatrix}$$

con

$$H_p = \begin{bmatrix} D & 0 & \dots & \dots & 0 \\ CB & D & 0 & \dots & 0 \\ CAB & CB & D & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ CA^{n-2}B & \dots & \dots & \dots & D \end{bmatrix} \quad O_p = \begin{bmatrix} C' \\ C'A \\ C'A^2 \\ \vdots \\ C'A^{n-1} \end{bmatrix}$$

$C' = [1 \ 0 \ 0]$  e  $T_i=T_i*ones(p,1)$ . Lo scopo di questo approccio è legare gli ingressi  $u$  all'andamento della temperatura nel sistema linearizzato semplicemente cambiando la matrice  $C$  con la matrice  $C'$  per imporre che il vettore  $y$ , che adesso rappresenta la temperatura, deve stare nel *range* desiderato.

Tuttavia questo comporterebbe un costo computazionale incompatibile con lo scopo di questo esercizio orientando la scelta verso un approccio analogo alla pianificazione con vincoli sull'ingresso: incrementare il numero di passi finché il vincolo non viene rispettato lavorando direttamente sul sistema non lineare. Questi approcci non sono equivalenti: il primo ha un vincolo sul numero di passi, il secondo no.

```

1 %% Pianificazione con vincoli sullo stato
2 T_max=1.5; %Limite superiore per la temperatura
3 T_min=-1.5; %Limite inferiore per la temperatura
4
5 while any([Tmax>T_max, Tmin<T_min, umax>u_max, umin<u_min])
6     Rp=Bd;
7     for i=2:p
8         Rp=[Bd, Ad*Rp];
9     end
10    u=pinv(Ad*Rp)*(x_f-Bd*u);
11    u=[up;u]; % Pianificazione ottima
12    timing=Ts*(0:size(u,1)-1)'; % Input per Simulink
13    uSim=[timing, flipud(u)]; % Input per Simulink
14    y_lin=lsim(sys, flipud(u), timing, x_0, 'zoh'); % Input per ...
        Simulink
15    ySim=[timing, y_lin]; % Input per Simulink
16    simOut=sim('Simulazione_sistema_nl','SaveOutput','on',...
17                'OutputSaveName','Tsim'); % Simulazione Simulink
18    Tsim = simOut.get('Tsim'); % Andamento temperatura
19    Tmax=max(Tsim);
20    %disp(Tmax);
21    Tmin=min(Tsim);
22    umax=max(u);

```

```

23      umin=min(u);
24      p=ceil(p*1.1);           % Incremento passi
25      %disp(p);
26  end
27  disp(['Trovato un tempo minimo di ' num2str(p*Ts)]);

```

Una volta eseguito questo ciclo, risulta che se la pianificazione viene fatta su  $p = 2669$ , corrispondenti a circa 133 secondi, i vincoli vengono rispettati.

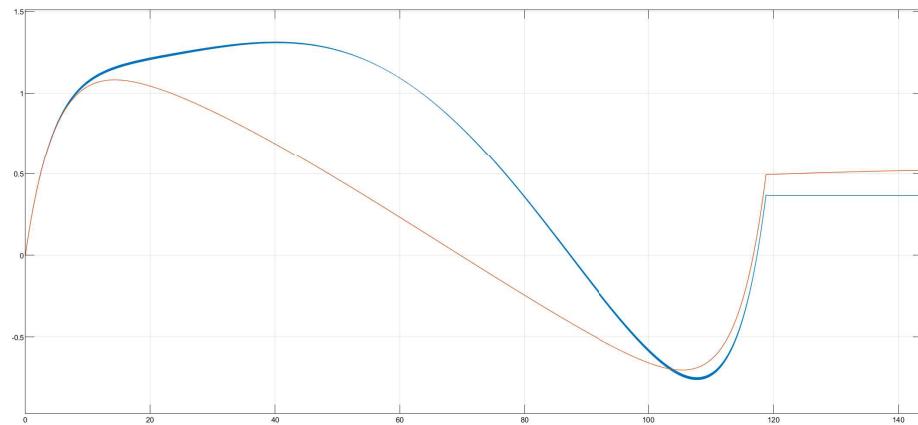


Figura 16: Andamento della temperatura: in rosso sistema non regolato, il quale diverge, in blu sistema regolato.

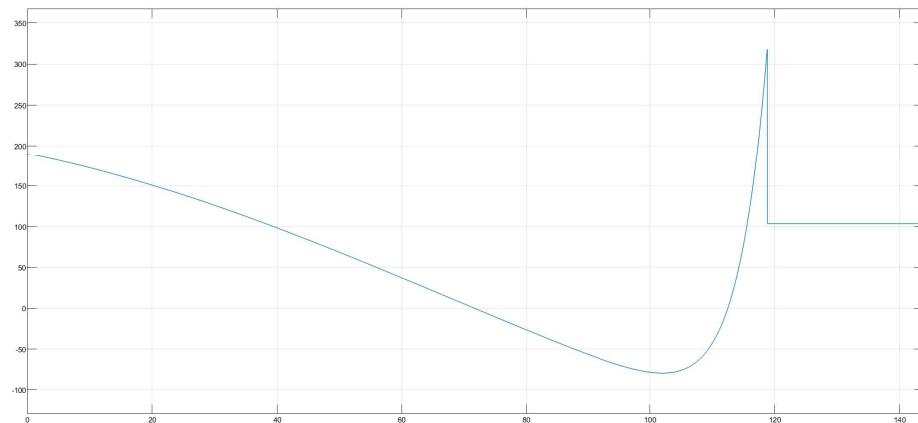


Figura 17: Ingressi con vincoli

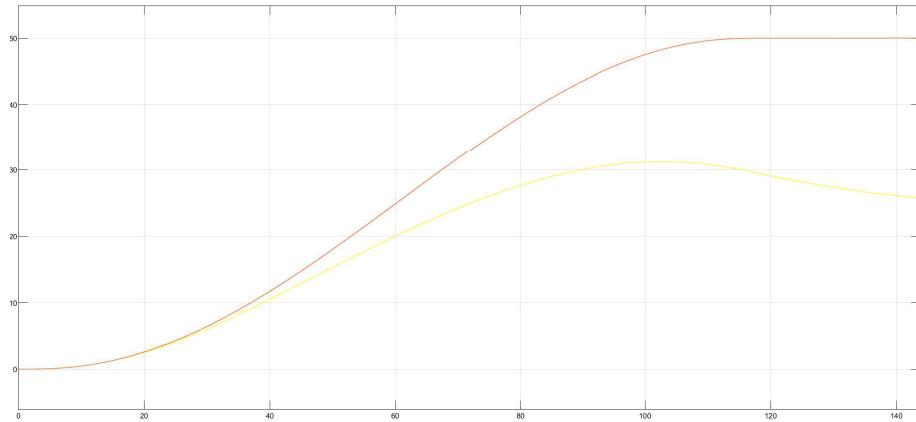


Figura 18: Andamento  $z$  nei tre sistemi senza regolatore

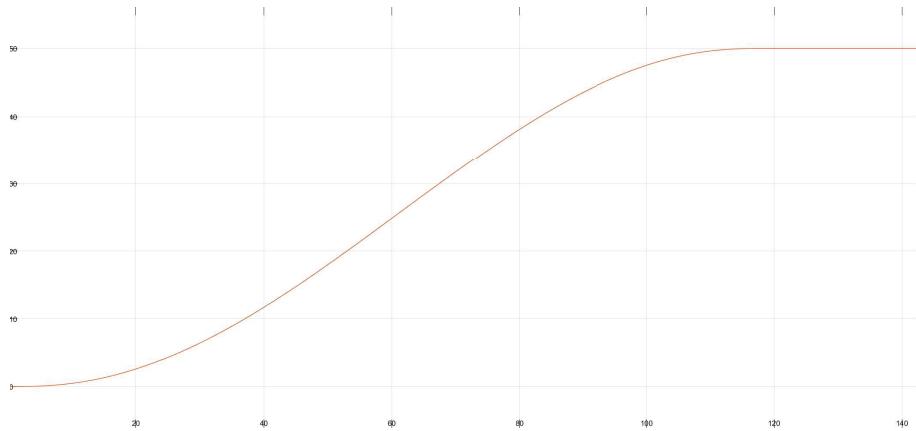


Figura 19: Andamento  $z$  nel sistema linearizzato e nel sistema non lineare con regolatore

## 5 Stima numerica della RAS

Nello studio della stabilità e nella sintesi del controllore è stato necessario linearizzare il sistema fisico nella forma dell'equazione 6. Così facendo viene commesso un errore

$$\begin{aligned}\tilde{f}(x, u) &= f(x, u) - Ax - Bu \\ \tilde{h}(x, u) &= h(x, u) - Cx = 0\end{aligned}$$

chiamato errore di linearizzazione. Tale controllore è montato in parallelo, per cui il sistema complessivo, adesso stabile, è esprimibile come

$$\begin{bmatrix} \mathbb{D}x \\ \mathbb{D}\tilde{x} \end{bmatrix} = \begin{bmatrix} A + BK & -BK \\ 0 & A + LC \end{bmatrix} \begin{bmatrix} x \\ \tilde{x} \end{bmatrix} + \begin{bmatrix} B \\ B \end{bmatrix} v + \underbrace{\begin{bmatrix} \tilde{f}(x) \\ 0 \end{bmatrix}}_{\tilde{f}(z)} \quad (11)$$

dove si intende  $\tilde{x} = x - x_r$  l'errore di stima.

Essendo il controllore montato sul sistema non lineare, occorre prestare attenzione: la funzionalità del controllore viene minata se ci si allontana troppo dal punto di equilibrio. Ciò fa nascere la necessità di capire fino a che punto l'approssimazione va bene ed esso, contestualmente, funziona. Infatti, indicando con  $z = [x \ x_r]^T$  lo stato complessivo, con  $Q$  una matrice simmetrica positiva definita e con  $P_Q$  la matrice soluzione dell'equazione di Lyapunov associata a  $Q$ , si può scegliere come candidata di Lyapunov  $V_Q = z^T P_Q z$  da cui

$$\dot{V}_Q = -z Q z^T + 2z^T P_Q \tilde{f}(z) \quad (12)$$

È evidente che nei pressi del punto di equilibrio, quindi con  $\tilde{f} \simeq 0$ ,  $\dot{V}_Q < 0$  e l'equilibrio è asintoticamente stabile mentre esisteranno valori di  $\tilde{f}$  che renderanno  $\dot{V}_Q \geq 0$ . Si vuole quindi capire in che *range* ci si può muovere senza che prevalga il termine di destra ossia senza alterare la stabilità del sistema.

Per il sistema in questione è possibile avere un'approssimazione numerica e cautelativa di tale regione utilizzando il teorema di Lasalle: viene adottato un algoritmo che ha come idea di base quella di provare un gran numero di candidate di Lyapunov  $V_Q$  e tracciare tutte le linee di livello compatibili con la stabilità del sistema. Per ogni candidata, vengono campionati tanti punti situati su una determinata curva di livello dal valore  $l$  per valutare in ognuno di essi  $\dot{V}_Q$ . Se la regione interna a tale curva di livello  $\Omega_l$  appartiene interamente alla RAS, tutti i punti avranno  $\dot{V}_Q < 0$ ; è sufficiente che uno solo abbia  $\dot{V}_Q \geq 0$  per poterla scartare. In seguito a questa valutazione si procede ad aggiornare il valore di  $l$  (se per tutti  $\dot{V}_Q < 0$  si aumenta il valore di  $l$  altrimenti si diminuisce) finché non viene raggiunto un limite fissato: a quel punto si cambia  $Q$  e si riesegue il tutto.

La stima viene fatta sul sistema complessivo per poi estrapolare solo le prime tre componenti dello stato complessivo vale a dire quelle associate al sistema vero.

Su Matlab questo può essere fatto utilizzando lo script seguente che serve a generare, ad ogni passo, una matrice  $Q$  casuale definita positiva per poi fare le valutazioni.

```

1 hFig=figure;                                % Inizializzazione figura
2 Rmin=0;                                     % Valore min curve di livello
3 Rmax=10;                                    % Valore max curve di livello
4
5 A_ras = [A_lin_eq, -B*K;
6     0, A_lin_eq+L*C];          % In alternativa Gc1.a
7
8 max_i=200;                                  % Numero di fun Lyap
9
10 for i=1:max_i
11     H=(rand(size(A_ras))-0.5);
12     Q=H.'*H;
13     Stima_ras_function(Q,A_ras,Rmax,Rmin,@f_tilde,hFig);
14 end

```

Il plot viene realizzato con la function `Stima_ras_function`, riportata in seguito, che serve a tracciare curve di livello compatibili con la stabilità per una determinata matrice  $Q$ , e quindi per una determinata candidata di Lyapunov.

```

1 N_STEPS=1e3;      %Numero di punti sulla curva di livello
2 P=lyap(A',Q);    %Soluzione dell'eq Lyapunov
3
4 M=inv(sqrtm(P));
5 figure(hFig);
6 hold on;
7 grid on;
8
9 while (Rmax-Rmin>0.0001)
10     R=Rmin+(Rmax-Rmin)/2;
11     zp=[];
12     for i=1:1:N_STEPS
13         x=rand(size(A,1),1)-0.5; %Generazione vettori casuali
14         y=sqrt(R)*x/norm(x);
15         z=M*y;                  %Proiezione sulla curva di livello
16         zp=[zp z];
17         vdot=-z'*Q*z+2*z'*P*fsys(z);
18         if vdot≥0
19             break;
20         end
21     end
22     if vdot<0           %Entra qui se la curva appartiene alla RAS
23         Rmin=R;
24         size(A,1)≥3
25         plot3(zp(1,:),zp(2,:),zp(3,:),'o');
26         xlabel('Temperatura');
27         ylabel('Quota');
28         zlabel('Velocita');
29         drawnow;
30     else                 %Entra qui se la curva non appartiene alla RAS
31         Rmax=R;
32     end
33 end

```

Balza all'occhio il diverso valore dell'estensione di tale luogo nelle tre direzioni: se per la velocità e la quota ci si può scostare di valori molto piccoli dalla condizione di equilibrio (inferiori in modulo a 0.15), la temperatura può raggiungere valori molto più elevati. Questo può essere spiegato con la migliore validità dell'approssimazione del sistema con il linearizzato per la temperatura rispetto alla quota e alla temperatura, come si evince dalle equazioni 3.

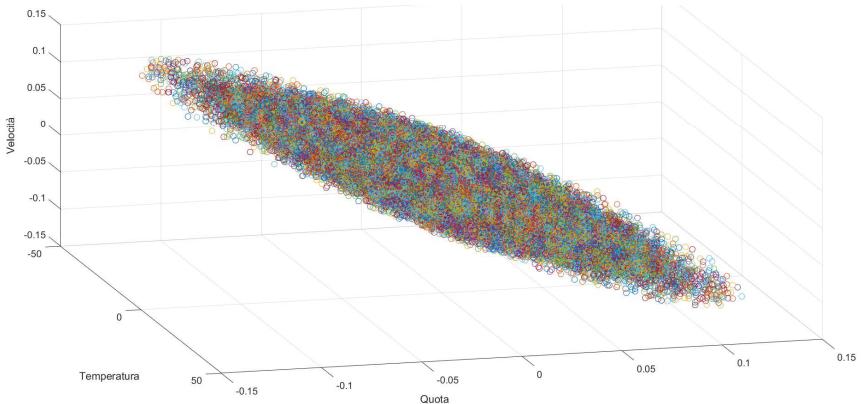


Figura 20: Visualizzazione nello spazio

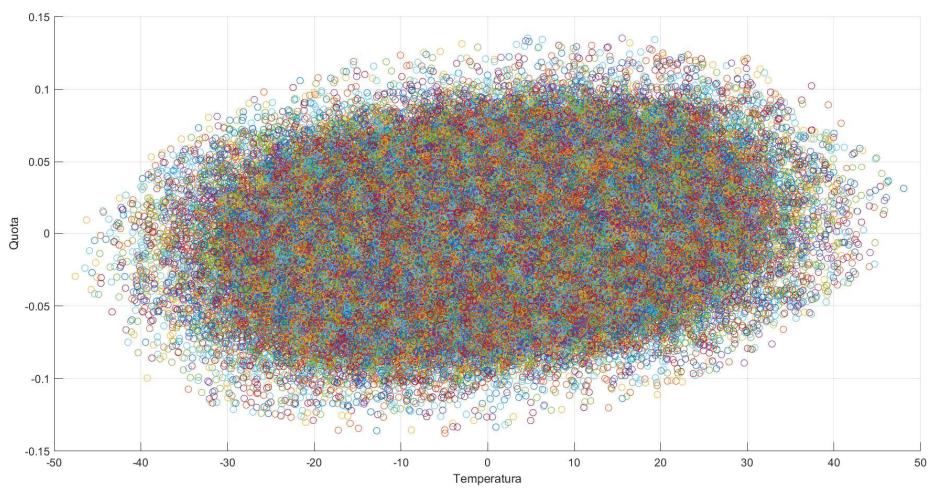


Figura 21: Visualizzazione sul piano  $T-z$

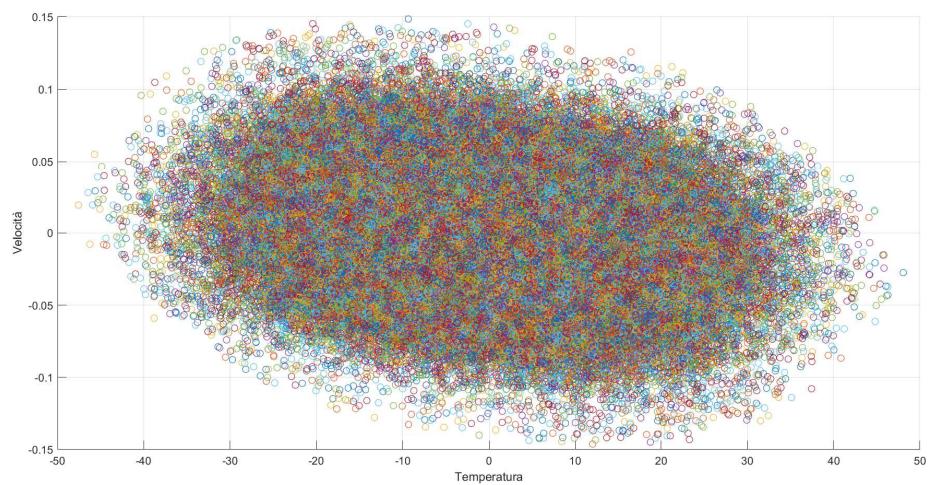


Figura 22: Visualizzazione sul piano  $T-v$

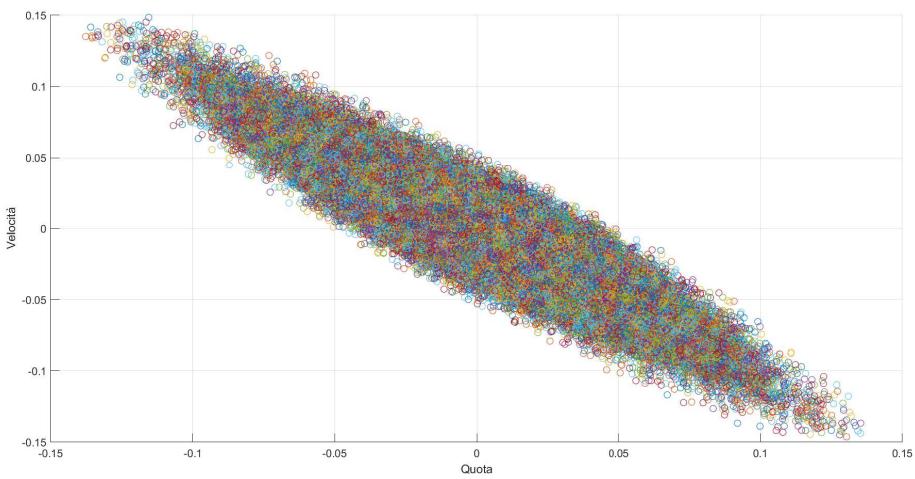


Figura 23: Visualizzazione sul piano  $v-z$

## Riferimenti bibliografici

- [1] Bicchi A., *Fondamenti di Automatica - I Parte*, 2015.
- [2] Bicchi A., *Fondamenti di Automatica - II Parte*, 2020.
- [3] International Standard Atmosphere, <http://fisicaatmo.at.fcen.uba.ar/practicas/ISAweb.pdf>.
- [4] Skyatch Personal Blimp specification, <http://www.personalblimp.com/specs.html>.