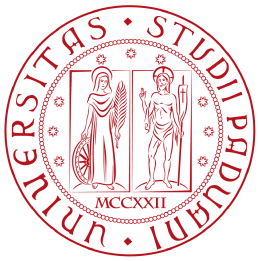

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria dell'Informazione

Tesi di Laurea Triennale

Progettazione, analisi e simulazione di un sistema di "cruise control" in MatLab/Simulink

Candidato
Marco Verri
1216512

Relatore
Prof.ssa Maria Elena Valcher

Anno Accademico 2021/2022
20 *settembre* 2022

Indice

Introduzione

Il Cruise Control è un sistema elettronico che si trova nella maggior parte dei veicoli moderni. Ha il compito di mantenere costante la velocità della vettura precedentemente scelta dal conducente, il quale può aumentare o diminuire la velocità impostata azionando due pulsanti. Inoltre se, ad esempio, il guidatore decidesse di sorpassare un'altra vettura premendo il pedale dell'acceleratore e aumentando la velocità, essa tornerà a quella impostata in precedenza solo quando si smetterà di accelerare.

Per funzionare esso fa uso di un sistema di controllo retroazionato che valuta la velocità corrente e la confronta con quella desiderata apportando le dovute modifiche alla potenza erogata dal motore, aumentandola o diminuendola in modo da raggiungere quella desiderata.

Tale dispositivo permette di avere vari vantaggi alla guida soprattutto dal punto di vista del comfort e dei consumi. Infatti il mantenimento della velocità costante consente di risparmiare carburante ed elimina le accelerazioni e decelerazioni dovute alla regolazione manuale della velocità.

Questo elaborato ha lo scopo di modellare e simulare il funzionamento del Cruise Control in modo che rispetti determinate condizioni di progettazione. Successivamente si procede all'analisi del sistema in catena aperta (risposta al gradino e diagrammi di Bode) e alla progettazione di un controllore attraverso diverse strategie (Controllori PID, luogo delle radici, rete ritardatrice). Infine si passa all'analisi in catena chiusa verificando il soddisfacimento delle condizioni iniziali.

Ogni passo terminerà con un'osservazione conclusiva al fine di confrontare i vari risultati.

Per tale studio si è utilizzato il software MatLab e Simulink.

Capitolo 1

Modellizzazione del Sistema

1.1 Configurazione fisica

Come già accennato nell'introduzione, il compito del Cruise Control è quello di mantenere costante la velocità del veicolo indipendentemente da fattori esterni come vento o inclinazione della strada. Ciò avviene misurando la velocità corrente, confrontandola con quella desiderata e regolando automaticamente l'accelerazione mediante una legge di controllo.

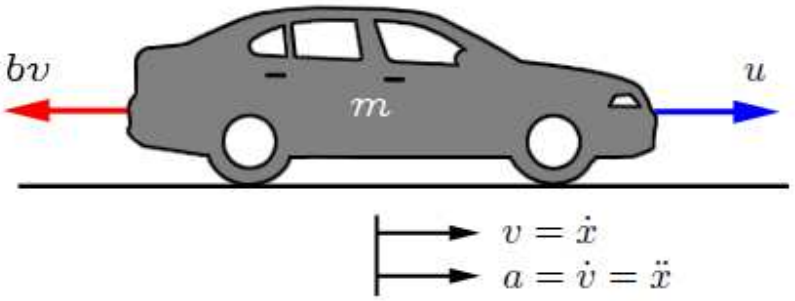

Figura 1.1: Modello grafico del sistema

In figura Fig. 1.1 è rappresentato il modello grafico del sistema. L'automobile, di massa m, è mossa da una forza di controllo u che rappresenta la forza generata dal motore che si trasferisce su strada tramite i pneumatici. Per semplificazione assumiamo di poter controllare questa forza direttamente non tenendo conto delle singole componenti che la generano. Le forze resistive bv, dovute dalla resistenza del vento e da quella degli pneumatici, variano linearmente con la velocità del veicolo v e agiscono in direzione opposta a quella del moto del automobile.
Definiamo ora le variabili di velocità (v) e accelerazione (a):

$$\begin{cases} v = \dot{x} \\ a = \dot{v} = \ddot{x} \end{cases}$$

1.2 Equazioni fisiche del modello

Con le assunzioni appena fatte possiamo dedurre che il sistema può essere rappresentato con un modello differenziale del primo ordine. Sommando tutte le forze che agiscono sull'asse delle ascisse e applicando la seconda legge di Newton ($F = ma$) arriviamo a definire la seguente equazione:

$$m\dot{v} + bv = u \tag{1.1}$$

Inoltre, visto che il fine ultimo è quello di controllare la velocità del veicolo, l'equazione dell'uscita sarà:

$$y = v \tag{1.2}$$

1.2.1 Parametri del sistema

Per continuare con l'analisi del sistema dobbiamo dare dei valori specifici alle incognite che compaiono nelle equazioni sopracitate. Tali valori sono riportati nella tabella qui di seguito:

Descrizione	Variabile	Valore
Massa	m	1000 Kg
Coefficiente di attrito	b	50 Ns/m
Forza di controllo	u	500 N

1.2.2 Modello in spazio di stato

I sistemi del primo ordine hanno un singolo modo di immagazzinare energia, in questo caso l'energia cinetica dell'automobile, pertanto serve solo una variabile di stato: la velocità. La rappresentazione in spazio di stato generale segue la forma:

$$\begin{cases} \dot{x}(t) = F \cdot x(t) + G \cdot u(t) \\ y(t) = H \cdot x(t) + J \cdot u(t) \end{cases} \tag{1.3}$$

dove:
F è una matrice quadrata di dimensione $n \times n$ (con n che rappresenta il numero delle variabili di stato),
G è una matrice di dimensione $n \times m$ (dove m rappresenta il numero di ingressi),
H è una matrice $p \times n$ (dove p rappresenta il numero di uscite),
J è una matrice $p \times m$ (che nel nostro caso è nulla).

Partendo quindi dalle equazioni 1.1 e 1.2 e rifacendoci alla forma generale, possiamo facilmente arrivare alla rappresentazione in spazio di stato:

$$\begin{cases} \dot{x} = \frac{-b}{m}x + \frac{1}{m}u \\ y = x \end{cases} \tag{1.4}$$

Per implementare il modello in MatLab dobbiamo inizialmente definire i parametri e le variabili del sistema come segue:

```
m = 1000;   %Massa del veicolo
b = 50;   %Coefficiente di attrito
u = 500;   %Forza di controllo
F = -b/m;
G = 1/m;
H = 1;
J = 0;
```

Infine utilizzando il comando *ss* si costruisce il modello in spazio di stato utilizzando le grandezze definite prima:

```
spazio_stato = ss (F,G,H,J); %Crea il modello in spazio di
    stato
```

1.3 Funzione di trasferimento

Dopo aver ricavato il modello in spazio di stato possiamo passare a definire la funzione di trasferimento. Facendo la Trasformata di Laplace dell'equazione differenziale 1.1 e assumendo le condizioni iniziali nulle otteniamo:

$$MsV(s) + bV(s) = U(s) \tag{1.5}$$

E sapendo che l'uscita del sistema è la velocità, la funzione di trasferimento diventa:

$$P(s) = \frac{V(s)}{U(s)} = \frac{1}{ms + b} \tag{1.6}$$

In MatLab:

```
s = tf('s');
F_t= 1/(m*s+b); %Definisco la funzione di trasferimento
```

1.4 Specifiche di prestazione

Il prossimo passo è definire delle specifiche di progetto che il sistema dovrà rispettare. Con i parametri che abbiamo definito possiamo notare che quando il motore eroga una forza di 500 N, il veicolo raggiungerà una velocità di picco di 11 m/s (22 mph - 36 km/h) in meno di 5 secondi. Con questi valori è accettabile, per la velocità, una sovraelongazione del 10% e un errore a regine del 2%.
Detto questo possiamo definire le specifiche di progetto come:

- Tempo di risalita < 5 s
- Sovraelongazione $< 10\%$
- Errore a regime $< 2\%$

Capitolo 2

Analisi del sistema in catena aperta

2.1 Risposta al gradino

La risposta al gradino è la risposta forzata di un sistema dinamico ad un segnale di ingresso $u(t) = \delta_{-1}(t)$.

La risposta al gradino del nostro sistema, senza nessun controllo in retroazione, ad un ingresso a gradino di ampiezza 500 N è simulata su MatLab come segue:

```
1 P_Cruise = 1/(m*s + b);
2 step(u*P_Cruise); % visualizza la risposta al gradino
```

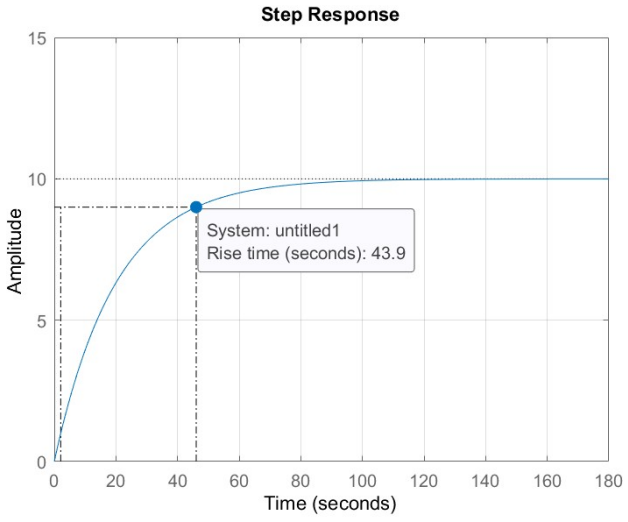

Figura 2.1: Risposta al gradino in catena aperta

Come possiamo notare la risposta al gradino non presenta alcuna sovraelongazione o oscillazione, dovuto dal fatto che è un sistema del primo ordine, inoltre raggiunge la velocità a regime desiderata di 10 m/s. Tuttavia il tempo di salita è

molto più lento di 60 s, quindi dobbiamo progettare un controllore in retroazione che velocizzi la risposta ma che non peggiori le altre dinamiche del sistema.

2.2 Studio di poli e zeri

Consideriamo ora la funzione di trasferimento calcolata prima 1.6, risulta immediato verificare che essa ha un unico polo in $s = -\frac{b}{m}$ che può essere mostrato in figura col seguente comando MatLab:

```
pzmap(P_Cruise); % visualizza sul piano complesso zeri e
    poli
axis([-1 1 -1 1]);
```

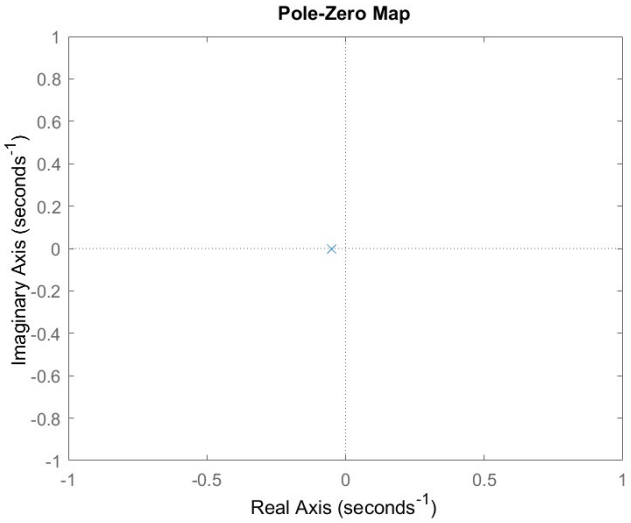

Figura 2.2: Grafico dei poli e zeri in catena aperta

Osserviamo che il polo è reale e negativo, quindi il sistema in catena aperta è stabile (BIBO-stabilità) e non oscilla. Inoltre la velocità di risposta è determinata dal modulo del polo ($\frac{b}{m}$): più grande è più il sistema e veloce a raggiungere il valore di regime. Non essendo noi in grado di alterare i parametri del nostro sistema per cambiare la risposta dinamica, dovremmo progettare dei controllori che modifichino la posizione degli zeri e dei poli del *sistema in catena chiusa* in modo da rispettare le performance desiderate.

2.3 Diagrammi di Bode

Passiamo ora allo studio della *risposta in frequenza* del sistema in catena aperta. Ciò può essere fatto graficamente attraverso i *diagrammi di Bode* delle ampiezze e delle fasi, in Matlab:

```
bode (F_t); %Traccia i diagrammi di Bode di modulo e fase
```

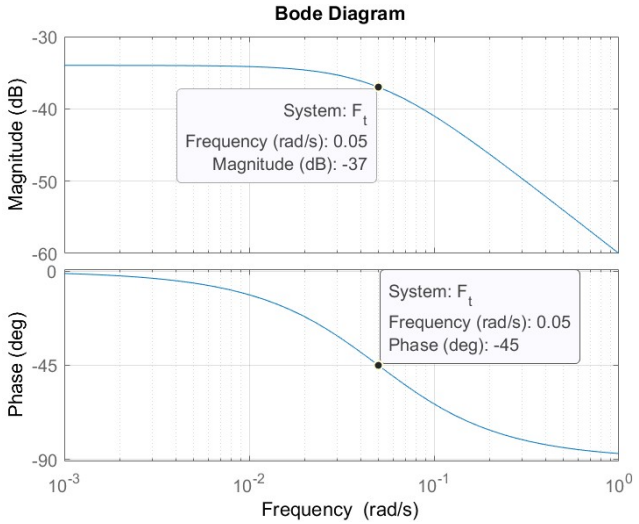

Figura 2.3: Diagramma di Bode di modulo e fase

Notiamo che i diagrammi di Bode ottenuti esibiscono il tipico comportamento dei sistemi del primo ordine, con il modulo di -3 dB e la fase di -45 deg in corrispondenza della pulsazione $\omega = \frac{b}{m} = 0.05$ rad/s. Inoltre, ad alte frequenze, il diagramma delle ampiezze diminuisce infinitamente con pendenza di -20 dB/dec.

Capitolo 3

Progetto del controllore

3.1 Breve introduzione

Un *controllore* è un dispositivo grazie al quale il progettista riesce a modificare le grandezze liberamente manipolabili (gli ingressi) all'interno di un sistema. In altre parole serve per far sì che un sistema retroazionato (in catena chiusa) riesca a soddisfare determinate specifiche che invece il sistema non retroazionato (in catena aperta) non rispettava.
Le principali specifiche di un progetto riguardano:

- **Stabilità BIBO**.
- **Errore a regime**: di quanto si differenzia il valore reale a regime da quello desiderato.
- **Tempo di assestamento**: il tempo necessario alla risposta per portarsi definitivamente a valori vicini al valore di regime.
- **Tempo di salita**: il tempo necessario al sistema per variare dal 10% al 90% del valore di regime dello stesso.
- **Sovraelongazione percentuale**: di quanto la risposta al gradino supera il valore a regime prima di assestarsi su di esso.
- **Attenuazione/reiezione dei disturbi**: capacità del sistema di smorzare l'effetto sull'uscita dei disturbi esterni.

Nel nostro caso il sistema può essere rappresentato come una retroazione unitaria negativa:

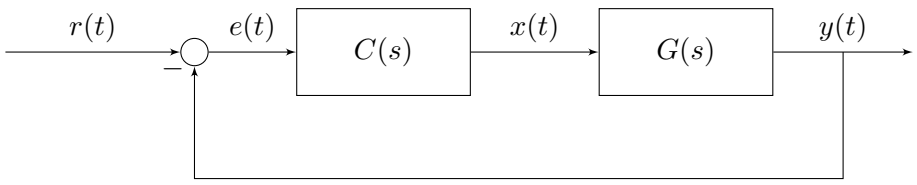

Figura 3.1: Schema a blocchi della retroazione unitaria negativa

dove:

- C(s) = funzione di trasferimento del controllore (quello che dobbiamo progettare noi).
- G(s) = funzione di trasferimento in catena aperta la cui espressione è data in (1.6).
- r(t) = ingresso del sistema.
- e(t) = r(t) - y(t) = errore di inseguimento.
- y(t) = uscita del sistema.

Possiamo semplificare unendo i due blocchi C(s) e G(s) sfruttando le proprietà della trasformata di Laplace:

$$\widetilde{G}(s) = C(s)G(s) \tag{3.1}$$

Calcoliamoci la funzione di trasferimento del sistema retroazionato:

$$W(s) = \frac{Y(s)}{R(s)} = \frac{C(s)G(s)}{1 + C(s)G(s)} = \frac{\widetilde{G}(s)}{1 + \widetilde{G}(s)} \tag{3.2}$$

Come si può facilmente notare dalla funzione di trasferimento appena ottenuta, il controllore C(s) compare sia al numeratore che al denominatore dando la possibilità al progettista di modificare il guadagno di Bode e di aggiungere poli e/o zeri al sistema .

3.2 Controllori PID

I controllori PID (Proporzionale-Integrale-Derivativo) sono una delle tipologie di controllori più usate data la relativa semplicità nel comprenderli, installarli e tararli. La funzione di trasferimento di un controllore PID è:

$$C(s) = K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s} \tag{3.3}$$

Possiamo notare dalla (3.3) che la funzione di trasferimento è composta dalla somma di tre componenti:

1. Parte **Proporzionale:** $C(s) = K_p$ riduce tempo di salita e errore a regime.
2. Parte **Integrale:** $C(s) = \frac{K_i}{s}$ elimina l'errore a regime per ingresso costante o a gradino unitario, ma può rallentare la risposta del sistema.
3. Parte **Derivativa:** $C(s) = K_d s$ aumenta la stabilità del sistema, riduce la sovraelongazione e migliora il comportamento transitorio.

Dal controllore PID si possono inoltre ottenere controllori più semplici variando i valori dei suoi parametri: controllore **PI** ($K_d = 0$), controllore **PD** ($K_i = 0$), controllore **I** ($K_d = K_p = 0$) e controllore **P** ($K_d = K_i = 0$).
In MatLab possiamo definire un controllore PID in questo modo:

```
Kp = 1; %Costante proporzionale
Ki = 1; %Costante integrativa
Kd = 1; %Costante derivativa
C = Kp + Ki/s + Kd*s
```

Altrimenti si può direttamente usare un comando MatLab che crea direttamente il controllore PID passandogli i valori dei suoi parametri:

```
C = pid (Kp, Ki, Kd)
```

3.2.1 Controllore P

Torniamo a considerare il nostro problema, come abbiamo constato in precedenza abbiamo bisogno di ridurre il tempo di salita del sistema per soddisfare le specifiche che ci sono state date. Per fare ciò utilizziamo un **controllore P** $C(s) = K_p$ sostituendolo nella nostra funzione di trasferimento (3.2):

$$W(s) = \frac{Y(s)}{R(s)} = \frac{C(s)G(s)}{1 + C(s)G(s)} = \frac{K_p \frac{1}{ms+b}}{1 + K_p \frac{1}{ms+b}} = \frac{K_p}{ms + b + K_p} \tag{3.4}$$

Creiamo ora il nuovo grafico della risposta al gradino ponendo $K_p = 100$ e velocità di riferimento di 10 m/s.
Il nuovo codice MatLab è:

```
m = 1000; %massa
b = 50; %coefficiente di attrito
r = 10; %velocita' di riferimento

s = tf ('s');
G = 1 / (m * s + b); %funzione di trasferimento G(s)

Kp = 100; %costante P
C = pid (Kp);

T = feedback (C * G, 1) %genera il sistema retroazionato
    negativamente

t = 0:0.1:20;
step (r * T, t)
axis ([0 20 0 10])
```

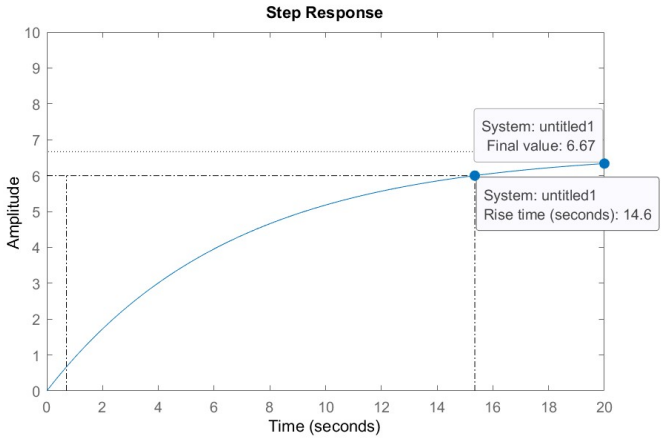

Figura 3.2: Risposta al gradino con $K_p = 100$

Come possiamo notare mettendo a confronto le figure 2.1 e 3.2, l'aggiunta del compensatore proporzionale ha velocizzato la risposta al gradino (che passa da 43.9 a 14.6 secondi), ma nonostante questo nè l'errore a regime nè il tempo di salita soddisfano le specifiche richieste.
Possiamo provare ad aumentare il guadagno proporzionale K_p portandolo ad un valore di 5000 per diminuire ancora di più il tempo di salita:

```
Kp = 5000;
C = pid(Kp);
T = feedback(C * G, 1);
step(r * T,t);
axis([0 20 0 12]);
```

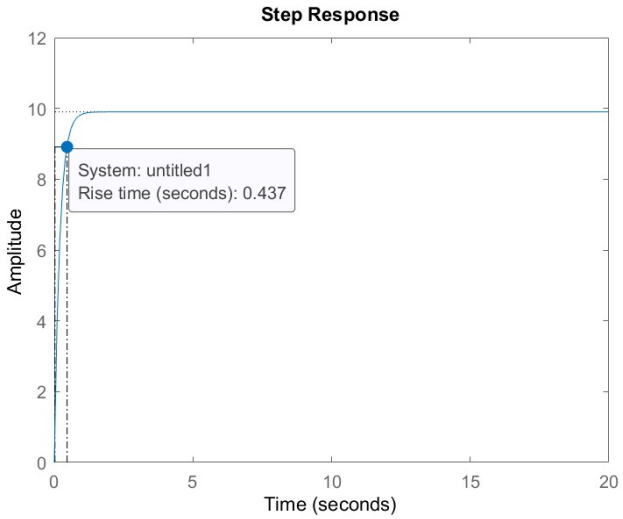

Figura 3.3: Risposta al gradino con $K_p = 5000$

Dalla figura 3.3 possiamo notare come l'errore a regime sia quasi nullo e il tempo di salita si sia ridotto notevolmente. Tuttavia questa risposta al gradino è **irrealistica** in quanto non esistono dispositivi di cruise control in grado di variare la velocità di un veicolo da 0 m/s a 10 m/s in meno di 0.5 secondi a causa della limitata potenza messa a disposizione dal motore.
La soluzione più corretta sarà quindi quella di diminuire il K_p, a costo di un aumento del tempo di salita, aggiungendo anche un controllore integrale (di tipo **I**) che permette di incrementare il tipo del sistema ed eliminare del tutto l'errore a regime.

3.2.2 Controllore PI

Come abbiamo già visto nella sezione 3.2 un controllore di tipo PI può essere ottenuto ponendo $K_d = 0$ nella formula generale dei controllori PID.
La sua funzione di trasferimento è quindi:

$$C(s) = K_p + \frac{K_i}{s} \tag{3.5}$$

Ugualmente a quello che abbiamo fatto per il controllore di tipo P sostituiamo l'equazione (3.5) nella funzione di trasferimento (3.2):

$$W(s) = \frac{Y(s)}{R(s)} = \frac{C(s)G(s)}{1 + C(s)G(s)} = \frac{(K_p + \frac{K_i}{s})\frac{1}{ms+b}}{1 + (K_p + \frac{K_i}{s})\frac{1}{ms+b}} = \frac{K_p s + K_i}{ms^2 + (b + K_p)s + K_i} \tag{3.6}$$

Come fatto prima ricaviamoci la risposta al gradino ponendo $K_p = 600$ e $K_i = 1$.
In MatLab:

```
Kp = 600;  %costante P
Ki = 1; %costante I
C = pid(Kp, Ki);
T = feedback(C * G, 1);
step(r * T,t);
axis([0 20 0 10]);
```

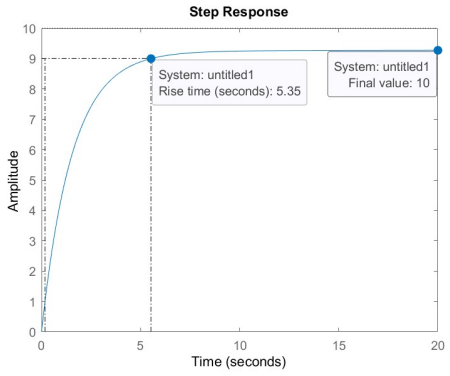

Figura 3.4: Risposta al gradino con $K_p = 600$ e $K_i = 1$

Dalla figura 3.4 notiamo che la specifica sul tempo di salita non è rispettata mentre, grazie all'introduzione del parametro integratore, l'errore a regime è nullo.

Bisogna quindi modificare i valori di K_p e K_i in modo da ottenere la risposta desiderata.
Per fare ciò bisogna prestare attenzione a non porre un K_i troppo elevato in quanto potrebbe indurre una sovraelongazione e un oscillazione non voluta.
Ciò può essere notato nelle risposte al gradino riportate nelle figure 3.5 e 3.6:

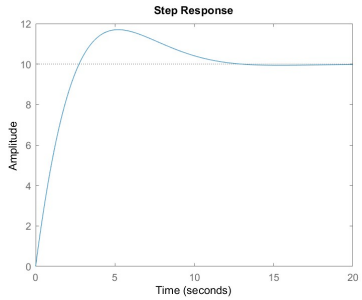

Figura 3.5: $K_p = 600$ e $K_i = 200$

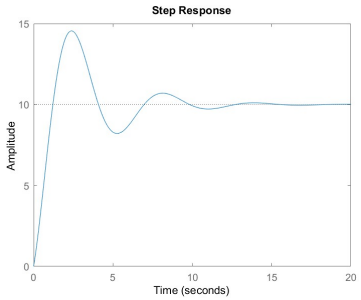

Figura 3.6: $K_p = 600$ e $K_i = 1300$

Una scelta ottimale per i valori di K_p e K_i è:

- $K_p = 800$
- $K_i = 40$

la risposta al gradino corrispondente è mostrata in figura 3.7:

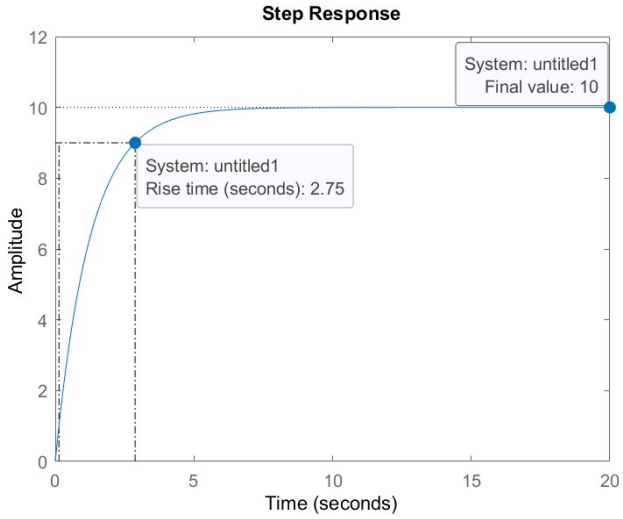

Figura 3.7: Risposta al gradino con $K_p = 800$ e $K_i = 40$

Il tempo di salita è di 2.75 secondi (soddisfa il vincolo progettuale), l'errore a regime è nullo (grazie all'integratore) e il tempo di assestamento è accettabile.

3.2.3 Controllore PID

Per questo specifico esempio, le specifiche di progetto sono state soddisfatte senza l'implementazione di un controllore derivativo. Tuttavia è utile osservare come l'ag-

giunta di quest'ultimo faccia variare il sistema.
Sostituiamo quindi la generica funzione di trasferimento in catena chiusa (3.2) con l'espressione che si ottiene in corrispondenza ad un controllore PID:

$$W(s) = \frac{Y(s)}{R(s)} = \frac{C(s)G(s)}{1 + C(s)G(s)} = \frac{K_d s^2 + K_p s + K_i}{(m + K_d)s^2 + (b + K_p)s + K_i} \tag{3.7}$$

Variando i valori dei parametri K_p, K_i e K_d cerchiamo di ottenere una risposta al gradino che soddisfi le specifiche di progetto.
Il codice MatLab è il seguente:

```
Kp = DA DETERMINARE %Costante P
Ki = DA DETERMINARE; %Costante I
Kd = DA DETERMINARE; %Costante D
C = pid(Kp,Ki,Kd);

T = feedback(C * G,1);
t = 0:0.1:20;
step(r * T,t);
axis([0 20 0 12])
```

Procedendo per tentativi arriviamo a determinare una possibile combinazione di valori che permette di soddisfare le specifiche sul tempo di salita, sulla sovraelongazione e sull'errore a regime: $K_p = 1000$, $K_i = 50$ **e** $K_d = 20$.
La risposta al gradino corrispondente è riportata in figura 3.8 e come si può notare il valore di regime è 10 m/s, il tempo di salita è 2.24 secondi e non sono presenti sovraelongazioni.

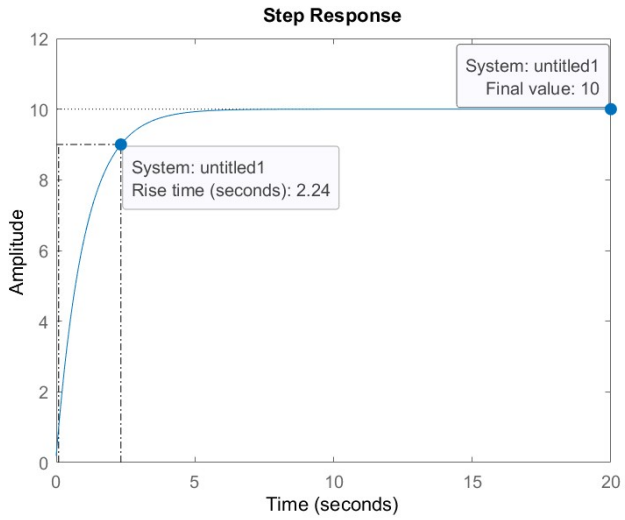

Figura 3.8: Risposta al gradino con $K_p = 1000$, $K_i = 50$ e $K_d = 20$

3.2.4 Osservazioni conclusive

A seguito delle analisi appena compiute possiamo affermare che l'uso di un **compensatore puramente proporzionale** permette di rispettare le specifiche di progetto solo teoricamente in quanto non è fisicamente realizzabile a causa della limitata potenza del motore.
Utilizzando un **controllore proporzionale integrativo**, invece, abbiamo visto come, scegliendo opportunamente i valori dei parametri K_p e K_i, possiamo ottenere delle soluzioni accettabili per il nostro problema.
Infine abbiamo mostrato come anche l'utilizzo di un **controllore PID** rende possibile il soddisfacimento dei vincoli progettuali, la funzione di trasferimento di quest'ultimo, però, è impropria e necessità quindi dell'inserimento di un polo ad alta frequenza il che rende più complicata realizzazione del controllore.
In conclusione quindi, a parità di prestazioni, è preferibile utilizzare un controllore di tipo PI rispetto a uno di tipo PID in quanto garantisce una maggiore semplicità costruttiva.

3.3 Analisi attraverso il luogo delle radici

3.3.1 Breve introduzione

Il **luogo delle radici** è il metodo grafico più utilizzato per verificare la stabilità di un sistema retroazionato, conoscendo i poli e gli zeri della funzione di trasferimento ad anello aperto. A differenza del metodo di Nyquist il luogo delle radici fornisce più informazioni specifiche sulla natura di ogni polo, tuttavia risulta meno robusto a variazioni parametriche del sistema rispetto agli altri metodi di studio.

Precedentemente abbiamo calcolato la funzione di trasferimento del sistema retroazionato (3.2):

$$W(s) = \frac{Y(s)}{R(s)} = \frac{C(s)G(s)}{1+C(s)G(s)}$$

con $C(s) = k_p$ (controllore proporzionale) e $G(s) = \frac{n(s)}{d(s)}$, l'equazione diventa:

$$W(s) = \frac{k_p n(s)}{d(s) + k_p n(s)} \tag{3.8}$$

Si noti che l'espressione appena ottenuta è rappresentazione irriducibile di $W(s)$.
Per studiare la stabilità del sistema bisognerà trovare i punti che annullano il denominatore (poli), quindi risolvere l'equazione:

$$d(s) + k_p n(s) = 0 \tag{3.9}$$

al variare di k_p.

3.3.2 Controllore proporzionale

Tornando al nostro problema, se sostituiamo l'espressione della risposta in catena aperta $G(s) = \frac{1}{ms+b}$ nell'equazione (3.8) otteniamo:

$$W(s) = \frac{k_p}{ms + (k_p + b)} \tag{3.10}$$

Per utilizzare i comandi MatLab che permettono di visualizzare il luogo delle radici, dobbiamo prima calcolarci il coefficiente di smorzamento (ξ) e la pulsazione naturale (ω_n), cioè passare dalle specifiche temporali a quelle in frequenza.
Per fare ciò, definito Tr il tempo di salita e Mp la massima sovraelongazione, quest'ultima può essere approssimata come (avendo compiuto un'approssimazione del primo ordine):

$$M_p \approx e^{\frac{-\pi\xi}{\sqrt{1-\xi^2}}} \tag{3.11}$$

Invertendo la formula e utilizzando i valori imposti dalle specifiche di progetto otteniamo:

$$\xi \geq \sqrt{\frac{\ln^2(Mp)}{\pi^2 + \ln^2(Mp)}} = 0.6 \tag{3.12}$$

Mentre per quanto riguarda la pulsazione naturale:

$$\omega_n \geq \frac{1.8}{Tr} = 0.36 rad/s \tag{3.13}$$

Ora possiamo passare all'implementazione in MatLab:

```
m = 1000;
b = 50;
r = 10;
s = tf('s');
P_Cruise = 1/(m*s + b);
rlocus(P_Cruise); %istruzione che grafica il luogo delle
    radici
axis([-0.6 0 -0.6 0.6]);
sgrid(0.6, 0.36); %coeff. di smorzamento e pulsazione
    naturale
```

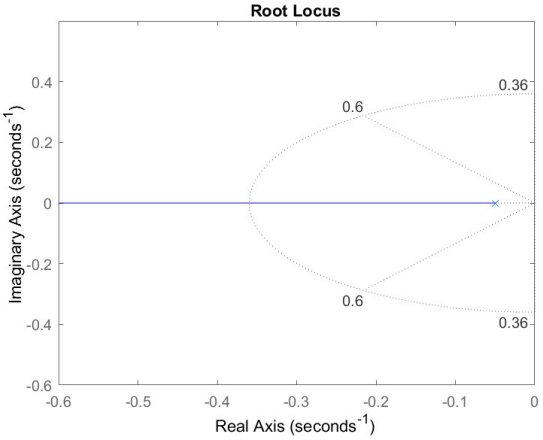

Figura 3.9: Luogo delle radici

Dalla figura 3.9 possiamo notare, come già avevamo detto in precedenza, che è presente un solo polo in -0.05, inoltre si possono notare varie linee tratteggiate. Le due linee inclinate rappresentano i punti del luogo in cui il coefficiente di smorzamento vale 0.6 (è maggiore nell'area compresa all'interno delle due linee e minore all'esterno), mentre il semi ellisse indica i punti del luogo in cui la pulsazione naturale vale 0.36 (è maggiore fuori da esso e minore all'interno).
Utilizzando il comando *rlocfind* è possibile selezionare un punto all'interno del luogo delle radici e come risposta ottenere il valore di k_p che deve avere la funzione di trasferimento in catena chiusa per avere un polo in tale punto:

```
[Kp, poles] = rlocfind(P_Cruise);
```

Dopo aver eseguito il comando, l'editor di MatLab chiederà di selezionare il punto nel quale si vuole il polo. Considerando le specifiche richieste dal problema, proviamo a selezionare un punto nell'intorno di -0.5:

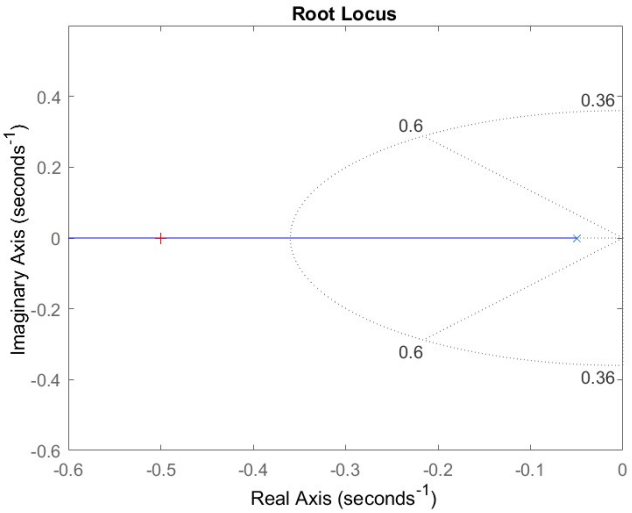

Figura 3.10: Luogo delle radici con polo selezionato

Dopo aver fatto ciò la Command Window ci darà la risposta:
selected_point =
-0.5000 - 0.0037i
Kp =
450.0154
poles =
-0.5000
Possiamo ora utilizzare il valore di k_p appena ricavato e inserirlo nella funzione di trasferimento per ottenere la seguente risposta al gradino:

```
Kp = 450.0154;
T = feedback(Kp*P_Cruise, 1);
t = 0:0.1:20;
step(r*T, t);
axis([0 20 0 11]);
```

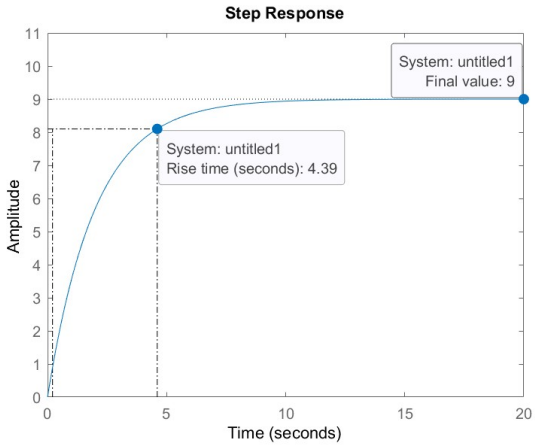

Figura 3.11: Risposta al gradino $K_p = 450.0154$

Dalla figura 3.11 notiamo che il vincolo sul tempo di salita (4.39 s) e sulla sovraelongazione sono rispettati, al contrario l'errore a regime è di circa il 10% che è maggiore del 2% richiesto dal progetto.
Ricaviamo comunque la risposta al gradino per diversi punti del luogo selezionati in modo da vedere come cambia man mano che ci si sposta lungo il ramo:

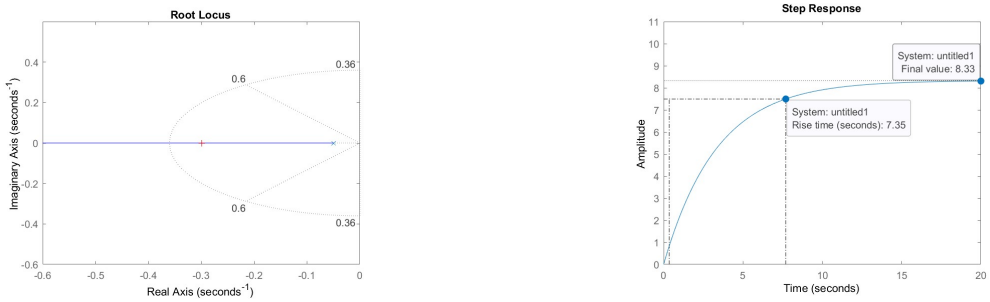

Figura 3.12: Punto selezionato -0.2991 - 0.0012i con $k_p = 249.0552$

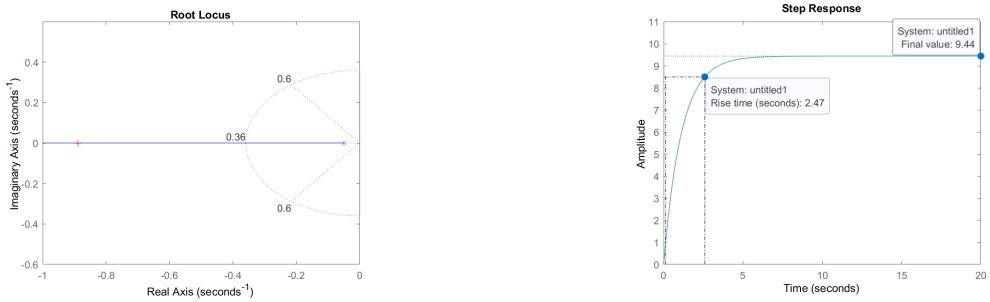

Figura 3.13: Punto selezionato -0.8886 - 0.0037i con $k_p = 838.6338$

Notiamo come l'allontanarsi del polo dall'origine lungo l'asse reale negativo corrispoda un aumento del guadagno K_p, questo fa si che sia il tempo di salita che l'errore a regime diminuiscono. Come abbiamo visto nel paragrafo 3.2.1 il controllore di tipo P presenta un limite legato alla realizzazione fisica in quanto la potenza erogata dal motore del veicolo è limitata, per ovviare a ciò bisogna scegliere un guadagno non troppo elevato e aggiungere al sistema una **rete ritardatrice**.

3.3.3 Introduzione di una rete ritardatrice (Lag controller)

Per ridurre l'errore a regime possiamo aggiungere al sistema una **rete ritardatrice**, la cui funzione di trasferimento è:

$$C_{lag}(s) = \frac{1 + \frac{s}{z_0}}{1 + \frac{s}{p_0}} \tag{3.14}$$

Con $0 < p_0 < z_0$.
Consideriamo ora la funzione di trasferimento del sistema retroazionato costituita dalla rete ritardatrice, dal controllore proporzionale e dalla funzione di trasferimento del sistema in catena aperta:

$$W(s) = \frac{k_p + k_p \frac{s}{z_0}}{\frac{m}{p_0}s^2 + (\frac{b}{p_0} + m + \frac{k_p}{z_0})s + (b + k_p)} \tag{3.15}$$

Scegliendo di posizionere il polo p_0 in $\omega_p = 0.03 rad/s$ e lo zero z_0 una decande dopo $w_z = 0.3 rad/s$ ed utilizzando il codice MatLab che segue graficamente si ottiene:

```
z0 = 0.3;
p0 = 0.03;

C_lag = (1+(s/z0))/(1+(s/p0)); % FDT rete ritardatrice

rlocus(C_lag * P_Cruise) % Rappresentazione grafica del
    luogo
axis([-0.6 0 -0.4 0.4])
sgrid(0.6,0.36); % Imposta i parametri di smorzamento e
    pulsazione
```

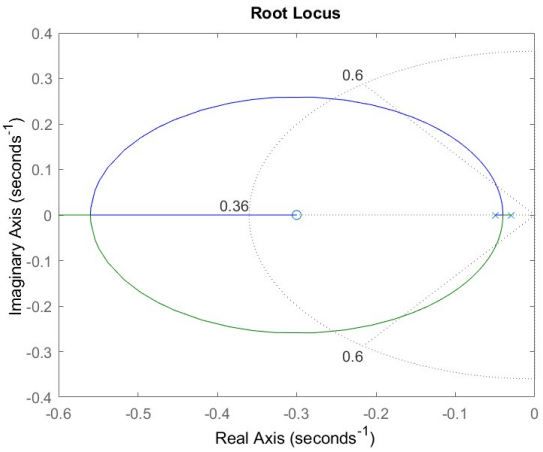

Figura 3.14: Luogo delle radici con rete ritardatrice

Come fatto in precedenza utilizziamo il comndo *rlocfind*:

```
1 [Kp, poles] = rlocfind(C_lag*P_Cruise);
```

per trovare il valore di k_p del punto selezionato. Scegliamo, per esempio, il punto -0.4 (la scelta del punto è stata fatta in modo che rispetti le considerazioni viste nel paragrafo precedente):

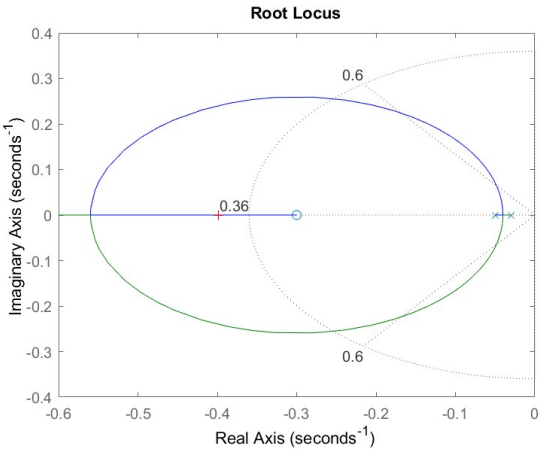

Figura 3.15: Punto selezionato

MatLab restituirà:

selected_point =

-0.3986 - 0.0008i

Kp =

1.3033e+04

poles =

-0.9847

-0.3986

Abbiamo ora il valore di k_p corrispondente al punto selezionato, calcoliamo ora la nuova risposta al gradino:

```
Kp = 13033;
T = feedback(C_lag*Kp*P_Cruise,1);
t=0:0.1:20;
step(r*T,t)
axis([0 20 0 12])
```

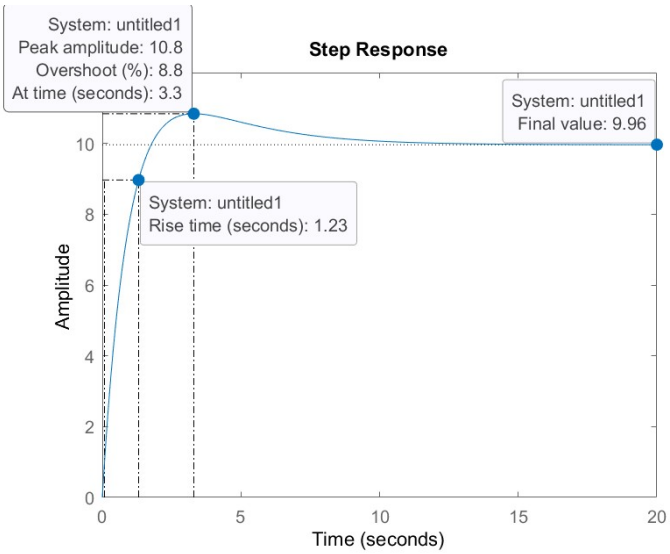

Figura 3.16: Risposta al gradino per punto selezionato

La rete ritardatrice ha fatto si che il sistema rispetti tutti i vincoli progettuali, sia sul tempo di salita che sull'errore a regime, l'aggiunta dello zero, però, ha introdotto una sovraelongazione che in questo caso risulta accettabile.

3.3.4 Osservazioni conclusive

Attraverso il metodo del **luogo delle radici** abbiamo potuto osservare ancora una volta come l'aggiunta di un solo guadagno proporzionale riduca sia l'errore a regime sia il tempo di salita con il rischio, però, di rendere il sistema non fisicamente realizzabile (conclusioni già discusse nel paragrafo precedente).
Per ovviare a tutto ciò abbiamo inserito una **rete ritardatrice** (coppia zero-polo con $\omega_p < \omega_n$) che ha fatto si che l'errore a regime diminuisse introducendo, però, una sovraelongazione accettabile in quanto compatibile coi vincoli progettuali.

3.4 Analisi in frequenza

3.4.1 Breve introduzione

Fino ad adesso abbiamo studiato il sistema in funzione di parametri temporali quali: tempo di salita, sovraelongazione e errore a regime.
Si può, però, passare ad analizzare il sistema nel dominio della frequenza, considerando cioè la **risposta in frequenza** $W(j\omega)$ ottenuta sostituendo $j\omega$ a s nella funzione di trasferimento.
Per compiere lo studio della risposta in frequenza dobbiamo introdurre dei nuovi parametri (ipotizzando la funzione di trasferimento $W(s) \in R(s)$ strettamente propria, bibo stabile e $W(0) \neq 0$):

- **Banda passante** B_p (a 3 dB): individua il più ampio intervallo di pulsazioni del tipo $[0, B_p]$ in cui si ha che $|W(j\omega)|_{dB} \geq |W(0)|_{dB} - 3dB$.
- **Pulsazione di risonanza** ω_r: se esiste, è quell'unica pulsazione positiva alla quale il modulo della risposta in frequenza assume valore massimo $|W(j\omega_r)| = max_{(\omega>0)}|W(j\omega)|_{dB}$.
- **Picco di risonanza relativo** M_{rel}: se la pulsazione di risonanza esiste, è definito come il rapporto tra il modulo della risposta in frequenza per $\omega = \omega_r$ e per $\omega = 0$, cioè $M_{rel} = [\frac{|W(j\omega_r)|}{|W(0)|}]_{dB}$.
- **Pulsazione di attraversamento** ω_a: se esiste ed è unica, è la pulsazione in corrispondenza della quale il diagramma di Bode del modulo espresso in dB si annulla (interseca l'asse delle ascisse): $|W(j\omega_a)|_{dB} = 0$.
- **Fase di attraversamento** φ_a: fase corrispondente alla pulsazione ω_a, ovvero $\varphi_a = arg(W(j\omega_a))$.
- **Margine di fase** m_φ: corrisponde alla grandezza $\varphi_a + \pi$.

3.4.2 Considerazioni sul sistema in catena aperta e in catena chiusa

Dalla funzione di trasferimento in catena aperta (1.6) possiamo calcolarci il guadagno a basse frequenze $K_B(\tilde{G})$ che nel nostro caso è $0.02 = -34dB$.
L'errore a regime si può calcolare in questo modo:

$$ss_{err} = \frac{1}{1 + K_B(\widetilde{G})} \cdot 100\% \tag{3.16}$$

I vincoli progettuali ci richiedono un errore a regime inferiore al 2%, per rispettarli dobbiamo ricavare un valore di $K_B(\widetilde{G})$ adeguato:

$$ss_{err} < 2\% \rightarrow \frac{1}{1+K_B(\widetilde{G})} \cdot 100\% < 0.02 \rightarrow K_B(\widetilde{G}) > 49 = 33.8dB$$

Dobbiamo quindi alzare il diagramma di Bode del modulo di almeno:

$$|33.8 - (-34)|_{dB} = 67.8dB = 2455$$

Imponiamo quindi un guadagno di 2500 e tracciamo il diagramma di Bode della funzione di trasferimento del sistema in catena aperta che otteniamo, evidenziando anche margine di fase e pulsazione di attraversamento (grazie al comando *margin*):

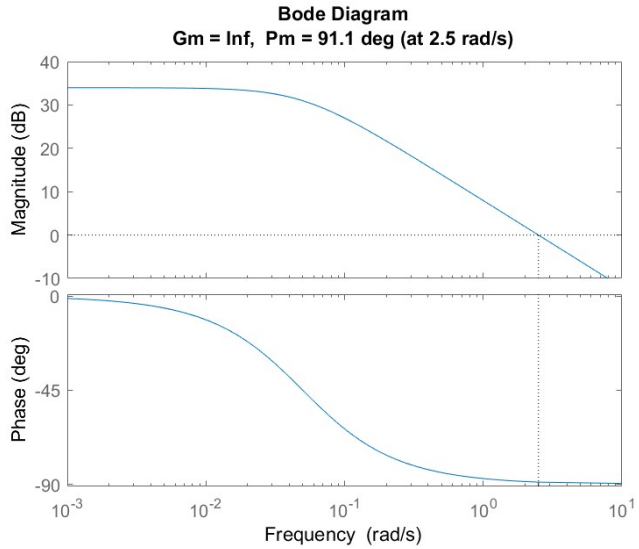

Figura 3.17: Diagramma di Bode con $K_p = 2500$

Possiamo notare che ora il guadagno a bassa frequenza è di 33.8 dB come richiesto, la risposta al gradino del sistema in catena chiusa dovrebbe quindi rispettare le specifiche. Verifichiamolo:

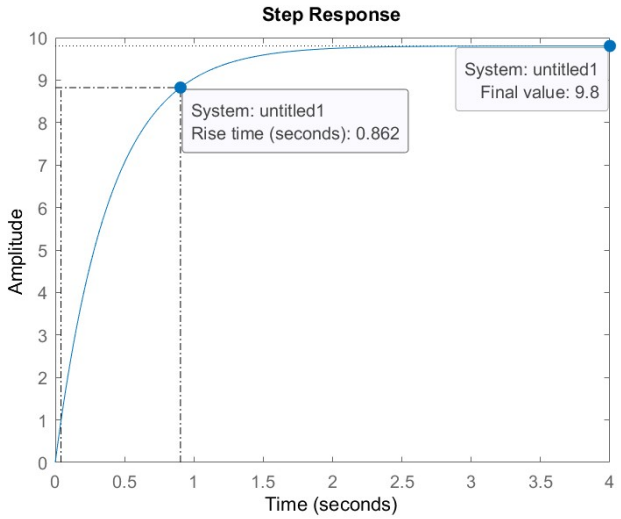

Figura 3.18: Risposta al gradino in catena chiusa con $K_p = 2500$

Per l'ennesima volta possiamo osservare come l'introduzione di un compensatore puramente proporzionale abbassi tempo di salita ed errore a regime in risposta a un gradino, rendendo però il sistema irrealizzabile fisicamente (un veicolo non può andare da 0 m/s a 9 m/s in meno di un secondo). Tutto ciò è in accordo con i

risultati riscontrati nelle analisi precedenti, per ovviare a tale problema utilizzeremo un guadagno minore inserendo una rete ritardatrice.

3.4.3 Introduzione di una rete ritardatrice (Lag controller)

Come già affrontato nelle sezioni precedenti l'introduzione di una **rete ritardatrice** permette di ridurre l'errore a regime e diminuisce la pulsazione di attraversamento rendendo il tempo di salita non troppo rapido.
La funzione di trasferimento è:

$$C_{lag}(s) = \frac{1 + \frac{s}{z_0}}{1 + \frac{s}{p_0}} \tag{3.17}$$

Sappiamo che l'errore a regime viene ridotto di un fattore $\frac{z_0}{p_0}$, poniamo quindi $z_0 = 0.1$ e $p_0 = 0.02$ in modo da ridurre l'errore di un fattore 5. Inoltre inseriamo anche un blocco proporzionale per diminuire il tempo di salita; poniamo, per esempio, $K_p = 3000$. Implementiamo il tutto in MatLab:

```
Kp = 3000;
z0 = 0.1; %scelta zero
p0 = 0.02; %scelta polo
C_lc = (1+(s/z0))/(1+(s/p0)); %FdT della rete ritardatrice
bode(Kp*C_lc*P_Cruise);
```

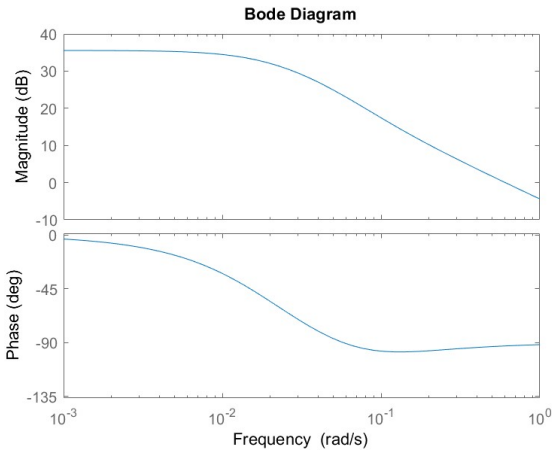

Figura 3.19: Diagramma di Bode del sistema in retroazione con rete ritardatrice

Verifichiamo che il sistema rispetti le specifiche richieste analizzando la risposta al gradino:

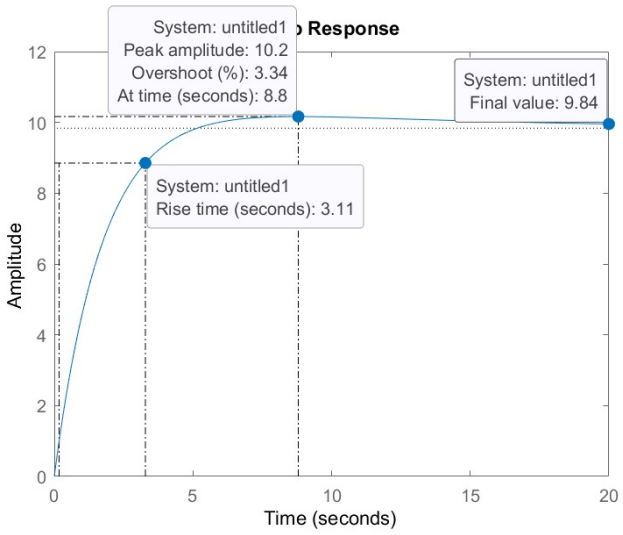

Figura 3.20: Risposta al gradino del sistema in catena chiusa con $K_p = 3000$ e rete ritardatrice

Possiamo notare come tutte le specifiche risultino soddisfatte: il tempo di salita è 3.11 secondi (< 5 s), la sovraelongazione è del 3.34% ($< 10\%$) e l'errore a regime è pari all'1.6% ($< 2\%$).

3.4.4 Osservazioni conclusive

Anche attraverso **l'analisi in frequenza** abbiamo riscontrato le stesse conclusioni dei paragrafi precedenti. Utilizzando un compensatore puramente proporzionale, infatti, rispettiamo tutti i vincoli progettuali ma andiamo incontro ad un problema di realizzabilità. Aggiungendo, invece, anche una rete ritardatrice, con opportuni parametri, riusciamo a rispettare tutte le specifiche di progetto.

Capitolo 4

Analisi mediante Simulink

4.1 Breve introduzione

Nel prossimo capitolo riprenderemo l'analisi fatta nel Capitoli precedenti implementando il tutto in Simulink, un estensione grafica di MatLab per la modellazione, simulazione e analisi di sistemi.
Grazie a questo strumento saremo in grado di rappresentare il modello tramite uno schema a blocchi (presi da una vasta libreria virtuale), rendendolo più facilmente comprensibile e analizzabile; inoltre Simulink offre anche la possibilità di simularne il funzionamento su un determinato intervallo temporale.
Prendendo quindi in considerazione il sistema già introdotto nei capitoli precedenti, con gli stessi parametri ($m = 1000\ Kg, b = 50\ \frac{Ns}{m}, u = 500\ N, r = 10\ \frac{m}{s}$) e le stesse specifiche di progetto, passiamo all'implementazione su Simulink.

4.2 Creazione del modello

Per modellare il sistema dobbiamo sommare tutte le forze che vengono applicate alla massa e integrare l'accelerazione per ottenere la velocità a cui il veicolo si sta muovendo.
Riprendiamo, quindi, l'equazione differenziale che rappresenta il sistema ricavata nel Capitolo 1:

$$m\dot{v} = u - bv \tag{4.1}$$

Dalla quale possiamo facilmente ricavare la formula per la velocità:

$$\dot{v} = \frac{u(t) - bv}{m} \rightarrow v = \int \frac{u(t) - bv}{m} dt = \int \frac{dv}{dt} dt \tag{4.2}$$

Cerchiamo quindi nella libreria "*continuous*" di Simulink il blocco **integratore** e rappresentiamo come ingresso l'accelerazione e uscita la velocità:

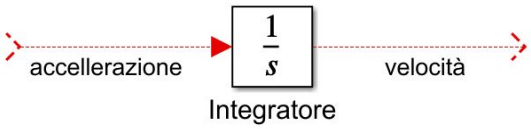

Figura 4.1: Blocco integratore

L'accelerazione ($\frac{dv}{dt}$) è data dalla somma delle due forze applicate al veicolo (u e bv) divise per la massa. Colleghiamo quindi un blocchetto "*gain*" (dalla libreria "*mathoperations*"), dalla parte di destra, all'integratore e diamogli il valore $\frac{1}{m}$:

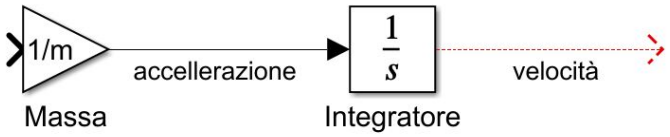

Figura 4.2: Blocco integratore e Gain

Colleghiamo ora al gain un blocco "*sommatore*" (stando attenti a modificare uno dei + in −) che servirà per effettuare l'operazione differenza tra $u(t)$ e $bv(t)$. Possiamo ora aggiungere le forze presenti nell'equazione (4.1):

- **Forza d'attrito**: inseriamo un ulteriore blocco "*gain*", con l'uscita collegata al blocco sommatore appena creato e l'ingresso all'uscita dell'integratore, e diamogli il valore b.
- **Forza d'ingresso**: attacchiamo, alla destra del sommatore, un blocco "*gradino*" (dalla libreria "*source*") che rappresenterà l'ingresso del sistema.

Infine, per visualizzare la velocità di uscita del sistema aggiungiamo un blocco "*scope*" (dalla libreria "*sinks*") e lo colleghiamo all'uscita dell'integratore:

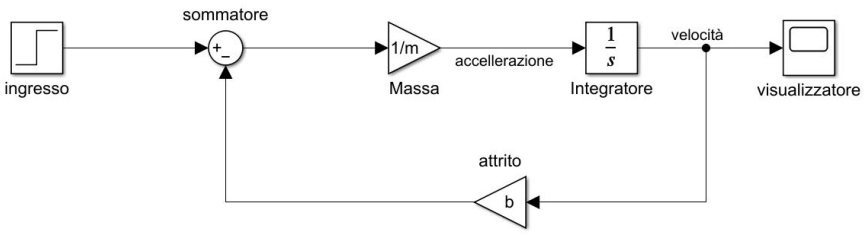

Figura 4.3: Schema a blocchi completo

4.2.1 Simulazione in catena aperta

Prima di far partire la simulazione modifichiamo i valori delle variabili del segnale "*gradino*" inserendo come valore finale u e come *step time* 0, per visualizzare la risposta al gradino nella sua interezza impostiamo la durata della simulazione a 120 secondi:

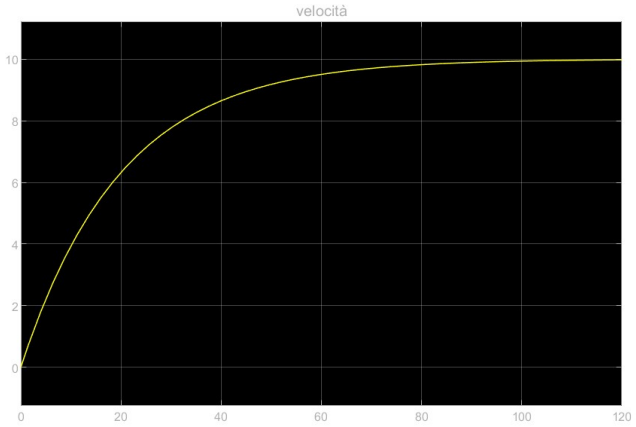

Figura 4.4: Risposta al gradino in catena aperta del sistema simulato

Come ci aspettavamo la simulazione ha prodotto la stessa risposta al gradino in catena aperta già analizzata nei capitoli precedenti (figura 2.1). Possiamo notare infatti come la risposta al gradino del sistema sia priva di errore a regime e di sovraelongazione ma presenti un tempo di salita molto maggiore dei 5 secondi richiesti dalle specifiche di progetto. È quindi necessario inserire un controllore per modificare il sistema in modo tale che rispetti i vincoli progettuali.

4.3 Simulazione in catena chiusa

Prima di proseguire con l'analisi in catena chiusa, possiamo racchiudere tutti i blocchi che fino ad ora abbiamo aggiunto al nostro sistema in un unico "sottosistema" in modo da rendere l'intero schema a blocchi più leggibile e "pulito".
Per fare ciò sostituiamo il blocco "*gradino*" e il blocco "*scope*" con, rispettivamente, un blocco "*In*1" e un blocco "*Out*1" (dalla libreria "*Ports & Subsystem*"):

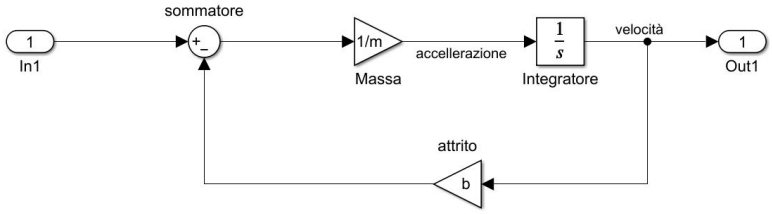

Figura 4.5: Schema a blocchi del sottosistema

Dopodiché possiamo inserire l'intero sistema in figura 4.5 all'interno di un blocco "*sottosistema*" (dalla libreria "*Ports & Subsystem*") e chiamarlo "sottosistema in catena aperta".

4.3.1 Controllore PI

Come già analizzato nei capitoli precedenti, per fare sì che il sistema rispetti le specifiche di progetto, è necessario utilizzare un controllore proporzionale integrale ovvero di tipo PI scegliendo opportunamente i suoi parametri.
Per "costruire" un controllore PI in Simulink dobbiamo:

- Aggiungere un bocco "*sommatore*" (modificando uno dei "+" in "-").
- Inserire due blocchi "*gain*" e attribuirgli ad uno il valore di K_i e all'altro quello di K_p.
- Posizionare un blocco "*integratore*" prima del blocco "*gain*" rappresentante K_i.

Infine sommiamo le due componenti del nostro controllore (K_i e K_p) in modo tale da ricavarne la funzione di trasferimento (3.5) e applichiamo all'ingesso un gradino e all'uscita un blocchetto "*scope*" per visualizzare graficamente la simulazione.

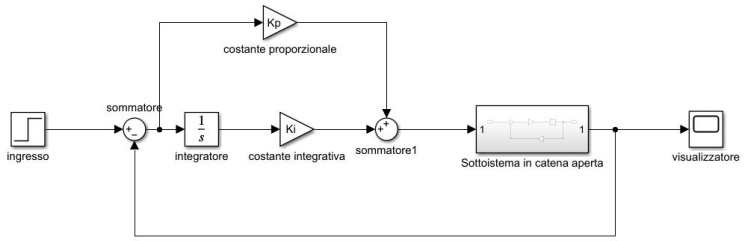

Figura 4.6: Schema a blocchi sistema in catena chiusa con controllore PI

In alternativa Simulink mette a disposizione un blocco "$funzione\ di\ trasferimento$" (dalla libreria "*Continuous*") per implementare il controllore nella sua interezza in modo più "pulito" e compatto. Basterà, infatti, fornirgli la funzione di trasferimento (3.5):

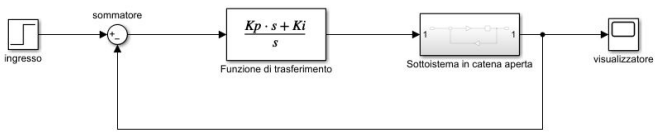

Figura 4.7: Schema a blocchi sistema in catena chiusa con controllore PI

Impostiamo, quindi, la durata della simulazione a 10 secondi e utilizziamo i valori del K_i e del K_p trovati nel paragarfo 3.2.2 che rendevano il sistema accettabile ($K_p = 800$ e $K_i = 40$).

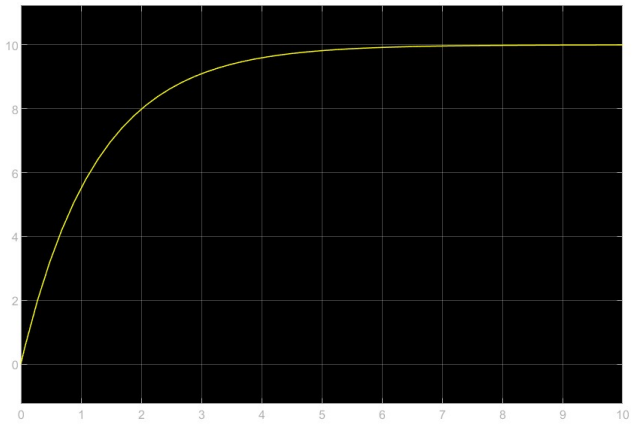

Figura 4.8: Risposta al gradino del sistema in catena chiusa con controllore PI

Come ci aspettavamo la risposta è identica a quella ottenuta nel paragrafo 3.2.2 (figura 3.7), si può quindi notare come tutte le specifiche di progetto siano rispettate (tempo di salita < 5 s, sovraelongazione $< 10\%$ e errore a regime $< 2\%$). Avendo già raggiunto gli obbiettivi prefissati con il controllore PI l'analisi mediante un controllore PID (paragrafo 3.2.3) non è necessaria.

4.3.2 Introduzione di una rete ritardatrice

Più interessante invece è l'implementazione di una rete ritardatrice in Simulink. Come già visto nei capitoli precedenti la **rete ritardatrice** permette di ridurre l'errore a regime del sistema retroazionato con controllore puramente proporzionale. Abbiamo già visto che la generica funzione di trasferimento è:

$$C_{lag}(s) = \frac{1 + \frac{s}{z_0}}{1 + \frac{s}{p_0}} \tag{4.3}$$

Per l'implementazione in Simlink basta inserire un blocco "$funzione\ di\ trasferimento$" contenente l'equazione 4.3, preceduto da un blocco "*gain*" contenente il valore della variabile proporzionale K_p.

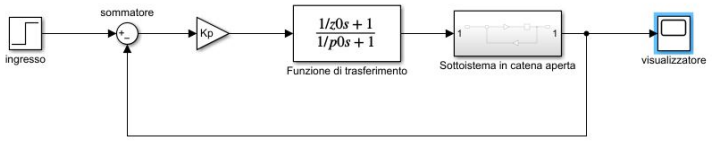

Figura 4.9: Schema a blocchi sistema retroazionato con rete ritardatrice

Impostiamo la durata della simulazione a 10 secondi e utilizziamo come valori delle variabili $z0$, $p0$ e K_p quelli ottenuti dall'analisi del luogo delle radici nel capitolo 3.3.3 ($K_p = 13033$, $z0 = 0.3$ e $p0 = 0.03$).

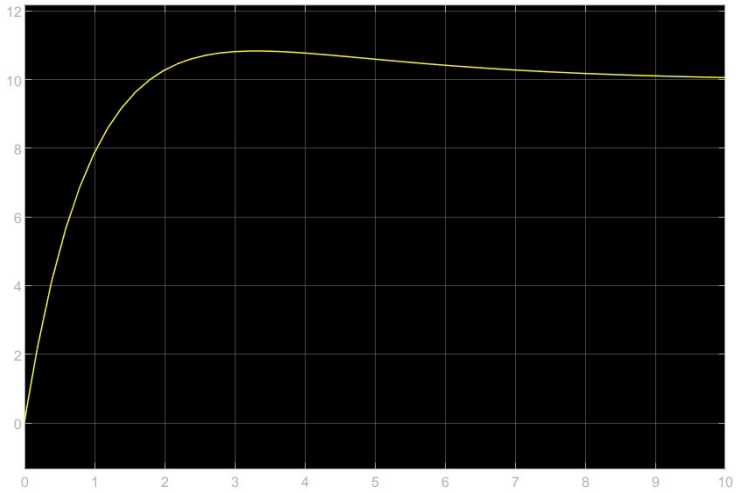

Figura 4.10: Risposta al gradino del sistema in catena chiusa con rete ritardatrice simulato

La risposta al gradino, come potevamo aspettarci, è uguale a quella ottenuta nel capitolo 3.3.3 (figura 3.16). Possiamo, quindi, trarre le stesse conclusioni: l'errore a regime è nullo e il tempo di salita rispetta il vincolo progettuale, inoltre la sovraelongaziuone che viene introdotta dall'aggiunta della rete ritardatrice è inferiore al 10% e quindi accettabile.

Conclusioni

Utilizzando l'ambiente MatLab e Simulink siamo riusciti a modellare, analizzare e simulare un sistema *cruise control* in modo che rispettasse i seguenti vincoli progettuali:

- **Tempo di salita $<$ 5 s**
- **Sovraelongazione $<$ 10%**
- **Errore a regime $<$ 2%**

Per compiere il nostro studio abbiamo, inoltre, fatto uso di diverse tecniche di modellizzazione di un sistema:

- Inizialmente abbiamo considerato il sistema in **catena aperta**.
- Abbiamo utilizzato **controllori di tipo P, PI e PID**.
- E infine è stata sfruttata una **rete ritardatrice (LAG)** sulla base dei risultati ottenuti con il *metodo del luogo delle radici* e con *l'analisi in frequenza*.

Possiamo riassumere i risultati ottenuti nella tabella seguente:

Controllore	Tempo di salita (s)	Errore a regime	Sovraelongazione
Nessuno	43.9	0	Assente
Tipo P	0.437	0,1%	Assente
Tipo PI	2.75	0	Assente
Tipo PID	2.24	0	Assente
LAG (Luogo delle radici)	1.23	0,4%	8.8%
LAG (Analisi in frequenza)	3.11	1,6%	3.34%

In rosso sono segnati i valori delle specifiche che non rispettano i vincoli di progetto.
Possiamo quindi concludere chentra tutti i metodi di modellizzazione che abbiamo utilizzato quello che risulta essere il migliore, anche dal punto di vista della semplicità progettuale, è l'utilizzo di un **controllore proporzionale-integrativo** (di tipo PI).

Bibliografia

[1] M. Bisiacco, M. E. Valcher, *"Controlli automatici"*, Libreria Progetto Padova, II Edizione, 2015

[2] Control Tutorials for MatLab&Simulink https://ctms.engin.umich.edu/CTMS/index.php?example=CruiseControlsection=SystemModeling