Table of Contents

1) Cruise Control.

Cos'è il Cruise Control.

Il **Cruise Control** è un sistema elettronico che consente ad un veicolo di mantenere una velocità costante impostata dal conducente, senza la necessità di agire costantemente sull'acceleratore.

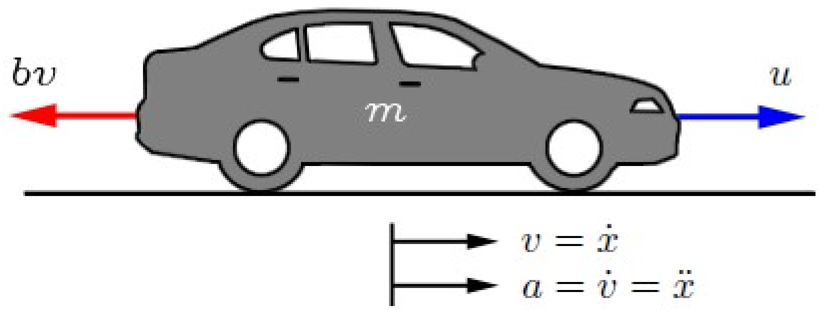
Principio di Funzionamento:

- **Impostazione della Velocità:** Il conducente imposta la velocità da mantenere, tramite pulsante o leva, e il sistema la memorizza come valore di riferimento.
- **Controllo della Velocità:** l'unità di controllo elettronico (**ECU**) monitora costantemente la velocità del veicolo tramite sensori (es, sensore di velocità). L'**ECU** confronta la velocità misurata con quella impostata: se è minore, l'ECU va ad agire sull'accelerazione per aumentarne la velocità, in caso contrario, si diminuisce l'accelerazione o si agisce sui freni.
- **Attuazione:** nel caso di motori a combustione, l'**ECU** agisce sull'acceleratore controllando elettronicamente l'iniezione del carburante e/o i freni, mentre per i veicoli elettrici si va ad agire direttamente sulla coppia (corrente) del motore. Per quest'ultimi, la gestione della decelerazione può essere gestita più facilmente sfruttando la frenata rigenerativa.
- **Disattivazione:** Il cruise control si disattiva automaticamente quando si preme il pedale del freno o anche tramite pulsante dedicato.

2) Analisi del modello Fisico.

Descrizione del modello Fisico

Il **Cruise Control** è un sistema a retroazione con lo scopo di controllare la velocità di crociera e mantenere costante la velocità del veicolo nonostante i disturbi esterni, come le variazioni del vento o della pendenza della strada. Ciò avviene misurando la velocità del veicolo, confrontandola conquella di riferimento e regolando automaticamente l'acceleratore in base a una legge di controllo.



Si considera un semplice modello della dinamica del veicolo, mostrato nel diagramma a corpo libero (immagine sopra). Sul veicolo, di massa **m**, agisce una forza di controllo **u,** che rappresenta la forza generata all'interfaccia strada/pneumatico. In questo caso si assume di poterla controllare direttamente trascurando l'attrito dei pneumatici, della pendenza della strada ecc. che contribuiscono a generare le forze resistive. Per questo modello si considera l'attrito viscoso bv (resistenza dell'aria) ipotizzando che vari linearmente con la velocità del veicolo v e che agisca in direzione opposta al moto. La forma linearizzata dell'attrito viscoso vale per basse velocità fintanto che il moto dell'aria risulta essere laminare.

Equazione del modello Fisico.

Un modo per descrivere i **Sistemi Complessi** è quello di analizzarli nel dominio del tempo e le grandezze che variano (nel tempo) possono essere definite dalle relazioni in termini di **Equazioni Differenziali**. La soluzione a queste equazioni fornisce il **legame ingresso-uscita** che dipende dal segnale in ingresso e dalle condizioni al contorno. Si ha quindi, per questo Sistema, che sommando le forze nella direzione **x** e applicando la seconda legge di Newton, si ottiene la seguente equazione:

$$\sum m\dot{v} = u - bv \rightarrow m\dot{v} + bv = u$$

che rapprsenta un'**Equazione Differenziale del primo ordine**, Lineare Tempo Invariante **(LTI)** --> **Descrivere meglio equazioni LTI.** Dal momento che si è interessati a controllare la **velocità** del veicolo, l'equazione di uscita può essere scritta come:

$$y = v$$

Descrizione del modello fisico con Equazioni di Spazio di Stato.

Data la precedente **Equazione Differenziale** e generalizzata in questa forma:

$$\sum_{i=0}^{n} a_i \frac{\mathrm{d}^i}{\mathrm{d}t^i} y(t) = u(t)$$

può essere descritta da un modello **Spazio di Stato**, che è una rappresentazione matematica di un sistema fisico come un insieme di variabili di input, di output e di stato correlate da **Equazioni Differenziali del Primo Ordine**. Le variabili di stato definiscono i valori delle variabili di output. Di seguito il modello nella forma standard:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$$

Dove:

- x : vettore di stato delle variabili (nx1).
- $\dot{x}$: derivata (rispetto il tempo) del vettore di stato (nx1).
- u : input o vettore di cotrollo (px1).
- y: vettore di output (qx1).
- A: matrice di sistema (nxn).
- B: matrice di input (nxp).
- C: matrice di output (qxn).
- D: matrice di feedforward (qxp)

In particolare la matrice di uscita C viene utilizzata per specificare quali variabili di stato (o combinazioni di esse) sono disponibili per l'utilizzo da parte del controllore. Dal momento che si è interessati a controllare la velocità del Sistema, si esplicita l'**Equazione Differenziale** in funzione di v:

$$\begin{cases} \dot{v} = -\dfrac{\mathrm{b}}{m} v + \dfrac{u}{m} \\ y = x \end{cases}$$

In questo caso il sistema **Cruise Control**, descritto da una sola **Equazione Differenziale**, lo si rappresenta tramite **Equazioni di Spazio di Stato**:

$$[\dot{v}] = \left[\frac{-b}{m}\right][v] + \left[\frac{1}{m}\right][u]$$

Il Tool di simulazione **Matlab**, permette di utilizzare la funzione **ss** per convertire modelli di sistemi dinamici nella forma di modello stato-spazio che risulta essere utile per Rappresentare un modello **Lineare Tempo Invariante (LTI)** ed eseguire la progettazione di controllo, che è lo scopo di questo elaborato.

```
% Preparazione script.
clc
clear
close all
```

```
% Dati del sistema.
m = 1000; % Massa del veicolo [kg].
b = 50;   % Coefficiente di attrito viscoso [Ns/m].
u = 500;  % Forza applicata al sistema [N].

% Matrici stato-spazio.
A = -b/m;
B = 1/m;
C = 1;
D = 0;

% Rappresentazione del sistema in forma di Spazio di Stato.
ss_cruise = ss(A,B,C,D)
```

```
ss_cruise =

  A =
            x1
   x1    -0.05

  B =
            u1
   x1    0.001

  C =
        x1
   y1    1

  D =
        u1
   y1    0

Continuous-time state-space model.
Model Properties
```

Descrizione del modello fisico con Trasformata di Laplace.

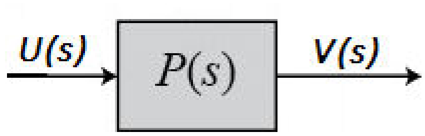
Per risolvere le **Equazioni Differenziali Lineari** risulta conveniente descrivere il problema di origine attraverso una trasformazione reversibile. Uno strumento che, dato un segnale di ingresso qualsiasi, fornisca calcoli semplici è dato dalla trasformazione dei sistemi dal dominio nel **tempo** a dominio **s**. Questo strumento è dato dalla **Trasformata di Laplace** dove la risoluzione di **Equazioni Differenziali** viene semplificata portando alla soluzioni di **Equazioni Lineari Algebriche**. La **Funzione di Trasferimento** di un sistema lineare è definito come il rapporto della trasformata di **Laplace** della variabile di uscita $Y(s)$ rispetto quella della variabile in ingresso $U(s)$, con tutte le condizioni iniziali poste a zero. La **FdT** di un generico sistema fisico è dato da:

$$F(s) = \frac{Y(s)}{U(s)}$$

che risulta essere un rapporto tra polinomi a coefficienti costanti e reali in **s**, dove il grado del polinomio al denominatore deve essere maggiore o uguale al grado del polinomio al numeratore.

Rappresentazione del modello fisico con Schemi a Blocchi.

In presenza di **SIstemi Complessi** risulta essere molto utile rappresentare la **Funzione di Trasferimento** tramite **Schemi a Blocchi**, dove l'azione causa-effetto è rappresentato da un blocco rettangolare e da un segmento entrate (ingresso) da uno uscente (uscita). Il **Cruise Control** è un sistema la cui FdT può essere descritto da il seguente schema:



dove si ha:

- $P(s)$: funzione di trasferimento.
- $U(s)$: ingresso.
- $V(s)$: uscita.

L' **Equazione Differenziale** che descrive il **Cruise Control** può essere riscritta in forma di Laplace come segue:

$$m\dot{v} + bv = u \rightarrow msV(s) \; + bV(s) = U(s)$$

si esplicita l'equazione rispetto la variabile di uscita e si porta la variabile di ingresso al denominatore. La **Fdt** del processo risulta essere:

$$V(s)(b + ms) = U(s) \rightarrow P(s) = \frac{V(s)}{U(s)} = \frac{1}{b + ms} \; \left[\frac{m/s}{N}\right]$$

e può essere riscritta in forma canonica:

$$P(s) = \frac{1}{b + ms} = \frac{K}{1 + \tau s} \quad \text{con} \begin{cases} K = \dfrac{1}{b} \\ \tau = \dfrac{m}{b} \end{cases}$$

Matlab mette a disposizione diversi strumenti per convertire modelli di **Sistemi Dinamici** in forma di funzioni di trasferimento:

- **ss2tf**: converte la rappresentazione dello spazio di stato in funzione di trasferimento.
- **tf**: rappresenta il modello della funzione di trasferimento.

```
% Conversione del sistema in funzione di trasferimento
[num,den] = ss2tf(A,B,C,D);

tf_cruise_ = tf(num,den)
```

```
tf_cruise_ =

   0.001
  --------
  s + 0.05

Continuous-time transfer function.
Model Properties
```

Rappresentazione in Forma Canonica.

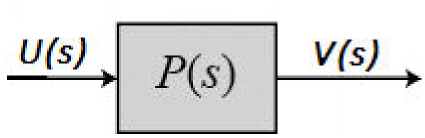
```
% Forma canonica.
K = 1/b;
Tau_p = m/b;

s = tf('s'); % Creazione variabile 's'.
P = K/(1 + Tau_p*s) % FdT del processo 'P' (Cruise Control).
```

```
P =

    0.02
  --------
  20 s + 1

Continuous-time transfer function.
Model Properties
```

3) Analisi del sistema.

Sistema a Catena Aperta.

Un controllore ad **Anello Aperto** è un sistema in cui l'uscita non viene misurata e utilizzata per regolare l'ingresso. Non ha quindi un meccanismo di feedback che gli permette di correggere eventuali errori o deviazioni rispetto l'uscita desiderata.



Stabilità del sistema.

La **Stabilita** del sistema è determinata dalla posizione dei **poli** della funzione di trasferimento nel piano complesso s. Un **Sistema Lineare Tempo-Invariante (LTI)** è stabile se e solo se tutti i **poli** della sua funzione di trasferimento hanno parte reale negativa e quindi si trovano nel semipiano sinistro del **piano s**. Per verificare questo si pongono i poli della **FdT** uguale a 0:

$$P(s) = \frac{1}{b + ms} \rightarrow b + ms = 0 \rightarrow s = -\frac{b}{m} \quad \text{con } b, m > 0$$

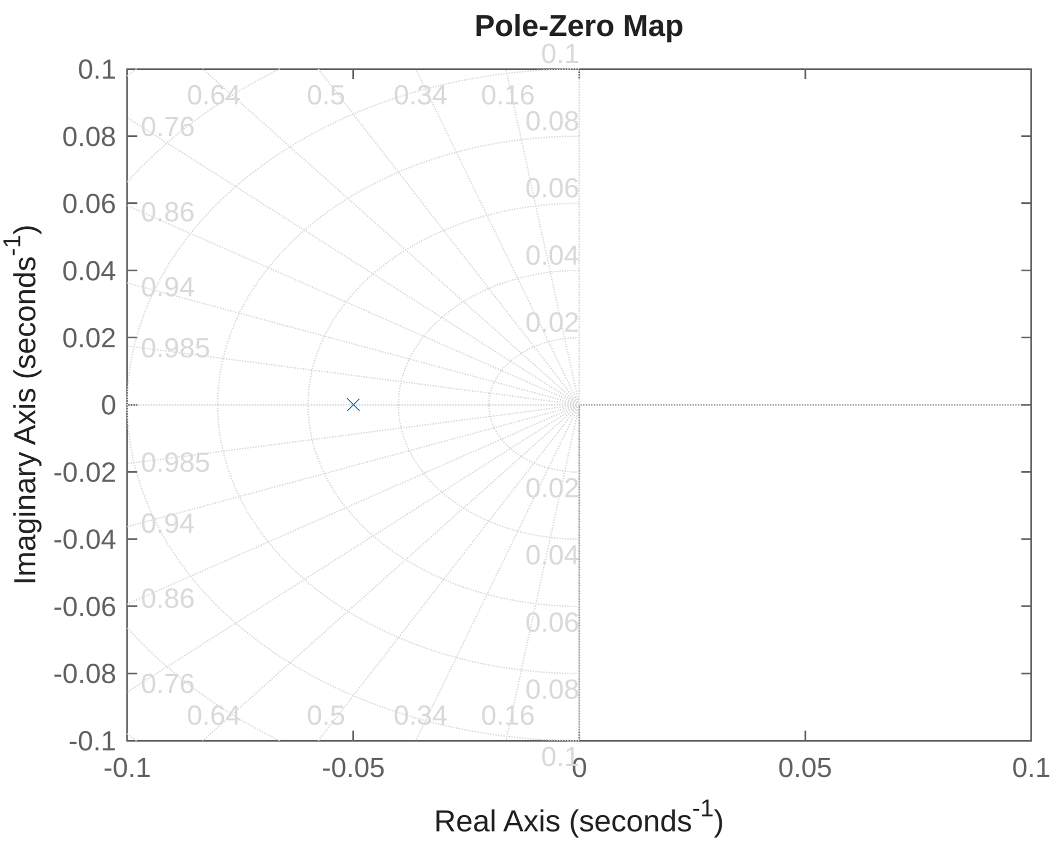
Il **Sistema** risulta avere un solo polo con parte reale negativa, quindi stabile. Per verificare queste condizioni, **Matlab** mette a disposizione diverse funzioni, alcune di queste sono:

- **pole**: restituisce i poli del sistema.
- **pzmap**: restituisce poli e zeri del sistema e li plotta.

```
% Poli del Sistema 'P'.
P_poli = pole(P)
```

```
P_poli =
-0.0500
```

```
% Mappa dei poli del Sistema 'P'.
pzmap(P);
axis([-0.1 0.1 -0.1 0.1])
grid on
```



Specifiche di prestazione.

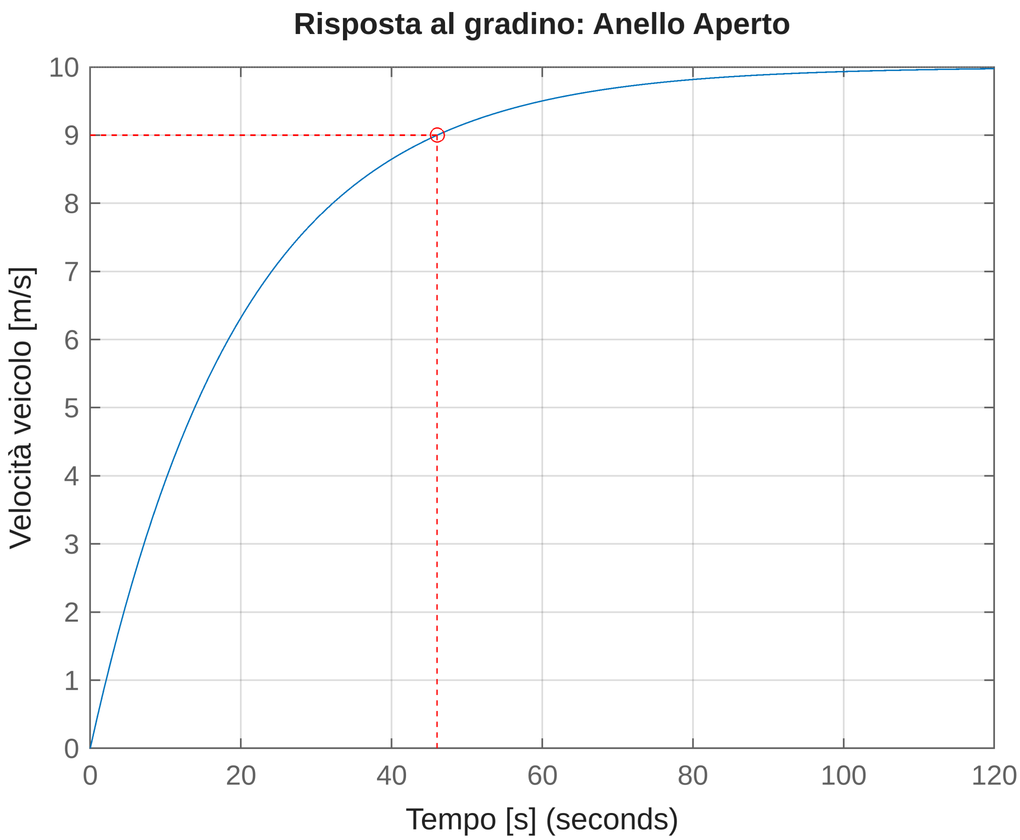
Dal momento che il **Sistema** è stabile, si definiscono le specifiche da soddisfare:

- Tr: Tempo di salita < 5 [s].
- Mp: Overshoot < 10 [%].
- e_{rr}: Errore a regime < 2 [%].

Si impone al sistema una forza **u** di 500 [N] per far raggiungere al veicolo una velocità ***v*** di 10 [m/s] (circa 36 [km/h]) dove si suppone che il veicolo sia in grado di raggingere quella velocità entro 5 [s] e il moto dell'aria ancora laminare (condizione per far valere la linearizzazione dell'attrito viscoso). Per verificarne le prestazioni, si stimola il **Sistema** con un segnale a gradino. Per ottenere questo, **Matlab** mette a disposizione le seguenti funzioni:

- **step**: risposta a gradino di un sistema dinamico.
- **RespConfig**: opzioni per la risposta a gradino.

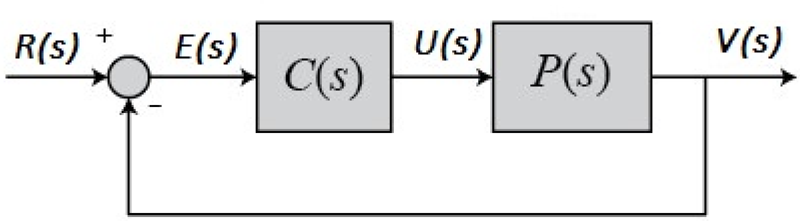
```
% Plot risposta a gradino.
respCfg = RespConfig('Amplitude',u);
[y,t] = step(P,respCfg);
stepplot(P,respCfg)

% Plot puntino rosso (Tempo di salita al 90%).
hold on
x_plot = t(51);
y_plot = y(51);
plot(x_plot,y_plot,'o','MarkerSize',5,'Color','r');
hold on

% Plot linee tratteggiate.
plot([0, x_plot], [y_plot, y_plot],'--','Color','r'); % Tratteggio
orizzonale.
hold on
plot([x_plot, x_plot], [0, y_plot],'--','Color','r'); % Tratteggio verticale.
grid on;
hold off;

% Legenda.
title('Risposta al gradino: Anello Aperto');
xlabel('Tempo [s]');
ylabel('Velocità veicolo [m/s]');
```



Dalla figura della **risposta a gradino** si nota che il **Sistema** non ha **Overshoot** (risultato aspettato in quanto sistema del primo ordine) e la risposta a regime raggiunge la valocità desiderata di 10 [m/s], ma il **Tempo di Salita** (90% del tempo totale) lo raggiunge dopo circa 46 [s]. Dal momento che quest'ultima specifica non è rispettata, risulta necessario progettare un controllore a **Catena Chiusa** che permetta di raggiungere le prestazioni desiderate, mantenedo la stabilità del sistema.

Sistema a Catena Chiusa.

Un controllore ad **Anello Chiuso**, detto anche sistema di controllo con **feedback**, è un sistema in cui l'uscita viene misurata e confrontata con l'ingresso desiderato (setpoint). La differenza tra questi due valori, chiamata errore, viene utilizzata per regolare l'ingresso del sistema, al fine di ridurre al minimo l'errore e far sì che l'uscita segua il setpoint.



dove si ha:

- $C(s)$: FdT del controllore.
- $P(s)$: FdT del processo.
- $R(s)$: ingresso.
- $E(s)$: errore.
- $U(s)$: uscita del controllore.
- $V(s)$: uscita.

Sintesi manuale del controllore.

Per ottenere le prestazioni desiderate, si utilizza un regolatore **PI** (Proporzionale-Integrativo) che è un algoritmo di controllo ad **Anello Chiuso** che combina un'azione **proporzionale** e un'azione **integrativa,** per regolare l'uscita di un sistema in modo da farla corrispondere a un valore desiderato (setpoint).Il **regolatore PI** introduce nella funzione di anello un **polo** nell origine e uno **zero** reale negativo, quindi è adatto a migliorare la precisione a regime dei sistemi di tipo 0 o 1, garantendo la stabilità. Il **PI** si comporta infatti, dal punto di vista della risposta in frequenza, come una rete ritardatrice con un polo nell origine. Pertanto, può migliorare il margine di fase per effetto dello **zero** ma determina una riduzione della larghezza di banda del sistema in anello chiuso. Data la funzione di trasferimento:

$$P(s) = \frac{1}{b + ms} = \frac{K}{1 + \tau_p s} \quad \text{con } K = \frac{1}{b}, \tau_p = \frac{m}{b}$$

Per questa applicazione si utilizza un **Regolatore PI** che cancelli il polo reale negativo e indicato con $C(s)$:

$$C(s) = PI = K_p + \frac{K_i}{s} = \frac{K_i}{s}\left(1 + \frac{K_p}{K_i}\right) = \frac{K_i}{s}(1 + \tau_c s) \quad \text{con } \tau_c = \frac{K_p}{K_i}$$

dove si ha:

- K_p: guadagno proporzionale.
- K_i: guadagno integrale.
- τ_c: costante di tempo del regolatore PI.

Si considera il sistema ad **Anello Aperto** $L(s) = C(s) \cdot P(s)$ e si provvede alla cancellazione **zero-polo**:

$$L(s) = C(s) \cdot P(s) = \frac{K_i}{s}(1 + \tau_c s) \cdot \frac{K}{1 + \tau_p s} = \frac{K_i K(1 + \tau_c s)}{s(1 + \tau_p s)}$$

dal momento che τ_p è un valore noto, è possibile cancellare il **polo** ponendo: $\tau_c = \tau_p$, si ottiene quindi:

$$L(s) = \frac{K_i K(1 + \tau_c s)}{s(1 + \tau_p s)} = \frac{K_i K}{s} \quad \text{con } \tau_c = \tau_p \rightarrow \frac{K_p}{K_i} = \frac{m}{b} = 20 \rightarrow K_p = 20 \cdot K_i$$

SI determina il valore di K_i studiando il sistema nel **dominio della frequenza** al valore della ω_c (omega critica) e quindi ponendo il modulo della funzione di trasferimento a catena aperta uguale a 1 (o a 0 dB, in scala logaritmica). In altre parole, è il punto in cui il guadagno del sistema, senza considerare il feedback, è unitario e dove si ha il massimo valore del **margine di fase**.

$$|L(j\omega_c)| = 1 \rightarrow \frac{K_i K}{\sqrt{(j\omega_c)^2}} = \frac{K_i}{b \cdot \omega_c} = 1 \rightarrow K_i = b \cdot \omega_c$$

Per un sistema del primo ordine è possibile calcolare la ω_c tramite la seguente relazione:

$$\omega_c = \frac{4.6}{T_s} \left[\frac{\text{rad}}{s}\right]$$

dove T_s è il tempo di assestamento del sistema con errore a regime dell' 1% e che quindi soddisfa anche il requisito dell'**errore a regime** e_{rr} del 2%. In questo caso si pone $T_s = 10\,[s]$ che è il tempo del sistema per arrivare a regime.

$$\omega_c = \frac{4.6}{T_s} = \frac{4.6}{10} = 0.46 \left[\frac{\text{rad}}{s}\right]$$

si sostituisce nella formula precedente e si calcola il valore di K_i e K_p:

$$\begin{cases} K_i = b \cdot \omega_c \rightarrow 50 \cdot 0.46 = 23 \\ K_p = 20 \cdot K_i = 460 \end{cases}$$

Per implemenare il controllore PI, Matlab metta a disposizione la funzione **pid** che permette di implementare il controllore **PI** passando rispettivamente i guadagni di di K_i e K_p come parametri in ingresso.

```
% Controllore PI.
Kp = 460; % Guadagno Kp.
Ki = 23;  % Guadagno Ki.
C = pid(Kp,Ki);
```

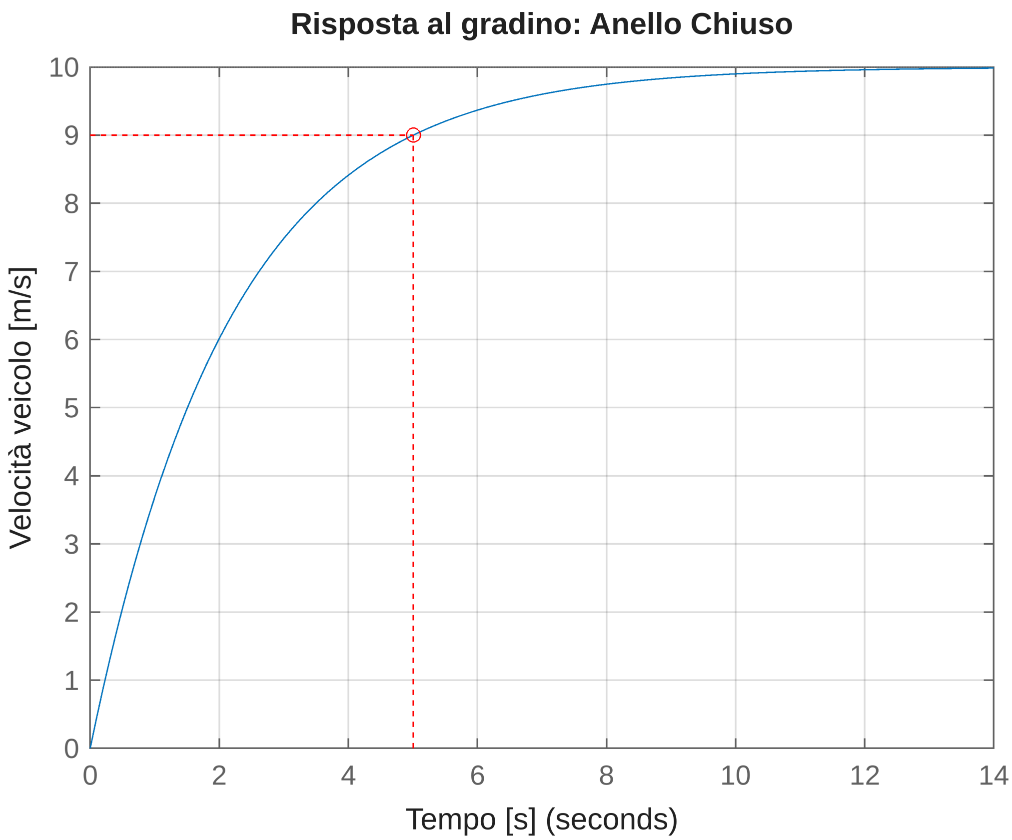
Una volta implementato il controllore $C(s)$ si calcola la **FdT** ad anello aperto $L(s)$ e la si pone nel sistema ad anello chiuso $W(s)$:

$$W(s) = \frac{L(s)}{1 + L(s)}$$

```
% Sistema ad anello aperto.
L = C*P;

% Sistema ad anello chiuso.
W = feedback(L,1);
```

Una volta determinata la funzione di trasferimento ad **anello chiuso**, si provvede a verificare se soddisfa i requisiti di prestazione scelti. Si utilizza la funzione **step** di Matlab, questa volta con un gradino di ampiezza $u = 10\,[m/s]$ in quanto in catena chiusa la variabile di feedback è la **velocità**, che è il valore da controllare.

```
% Plot risposta a gradino.
u = 10; % Ampiezza del gradino. [m/s].
respCfg = RespConfig('Amplitude',u); % Configurazione del gradino.
[y,t] = step(W,respCfg);
stepplot(W,respCfg)

% Plot puntino rosso (Tempo di salita al 90%).
hold on;
x_plot = t(51);
y_plot = y(51);
plot(x_plot,y_plot,'o','MarkerSize',5,'Color','r');
hold on;

% Plot linee tratteggiate.
plot([0, x_plot], [y_plot, y_plot],'--','Color','r'); % Tratteggio
orizzonale.
hold on;
plot([x_plot, x_plot], [0, y_plot],'--','Color','r'); % Tratteggio verticale.
grid on;
hold off;

% Legenda.
title('Risposta al gradino: Anello Chiuso');
xlabel('Tempo [s]');
ylabel('Velocità veicolo [m/s]');
```

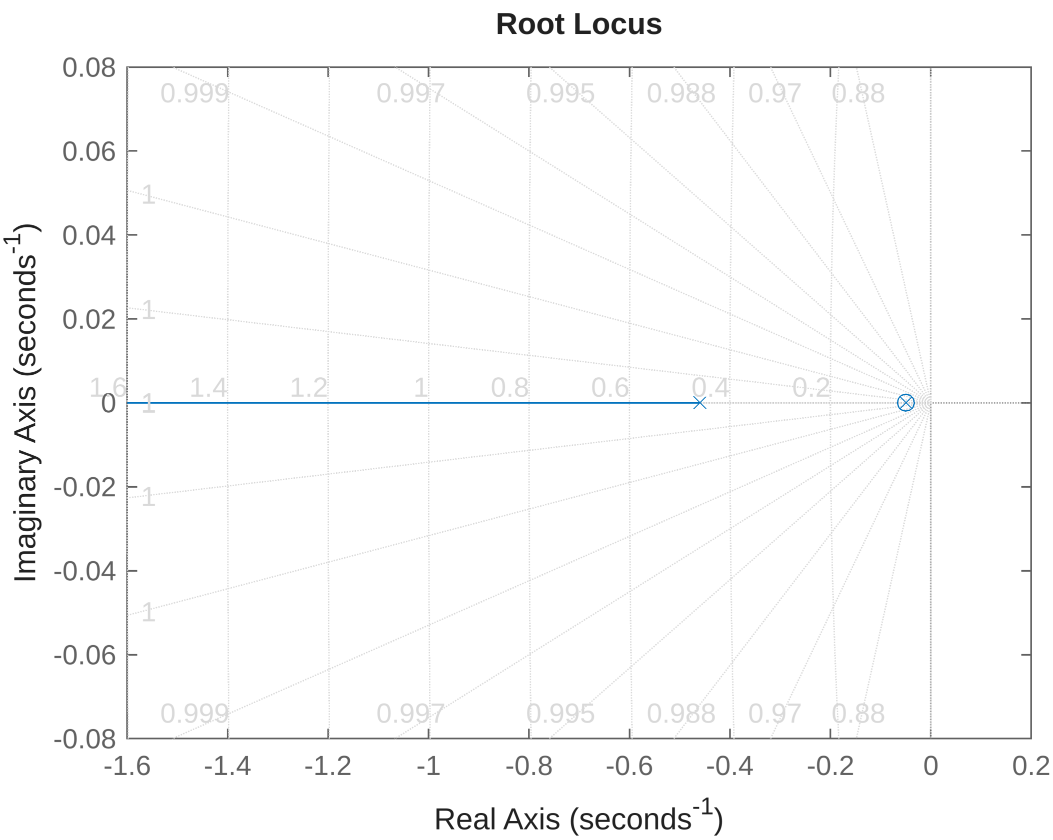


Si verifica la stabilità del sistema a catena chiusa:

```
% Poli del Sistema 'P'.
W_poli = pole(W)
```

```
W_poli = 2×1
   -0.4600
   -0.0500
```

```
% Luogo delle radici.
rlocus(W)
grid on;
```



Anche in questo caso il sistema risulta essere stabile in quanto tutti i poli hanno parte reale negativa. Come ci si poteva aspettare si hanno:

- un **polo** ed uno **zero** reali e coincidenti con $\omega_c = \omega_p = 0.05$ [rad/s].
- un **polo** con $\omega = 0.46$ [rad/s].

Osservando il **luogo delle radici** si nota che il polo situato a $p = -0.46$ tende a $-\infty$ all'aumentare di s e quindi stabile.

Sintesi del controllore con Matlab pidTuner.

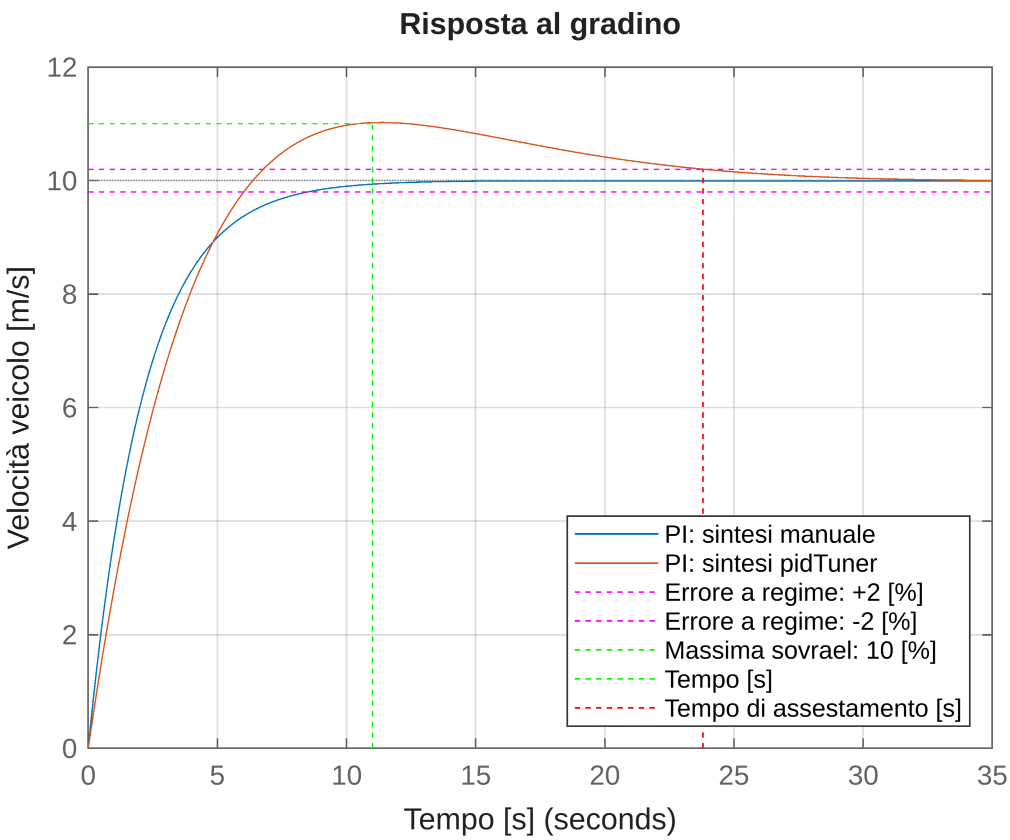
L'app **PID Tuner** permette di sintetizzare automaticamente i guadagni di un controllore **PID** con l'obiettivo di raggiungere un equilibrio tra prestazioni e robustezza e quindi migliorare le prestazioni del controller in modo interattivo. È possibile specificare il tipo di controller come **PI, PD, PID**, ecc dove i grafici permettono di esaminare le prestazioni del controller nel dominio del tempo e della frequenza. Il Sistema da indicare nella funzione **pidTuner** è il processo $L(s)$ a catena aperta.

Nell'app **PID Tuner,** indicando i valori di **Transient Time** a 4.35 [s] e **Transient Behaviour** a 0.9 si ottengono gli stessi risultati della taratura manuale, a conferma dei calcoli effettuati manualmente. Sempre con l'app di **Matlab** si provano a calcolare altri valori di K_p e K_i che rendano il **Tempo di Salita** più lineare e quindi più idone ad un'applicazione reale. Dopo alcuni tentativi, impostando nei due parametri precedenti, rispettivamente i valori 5.943 [s] e 0.756, si ottengono $K_p = 313.7$ e $K_i = 44.5$.

```
% pidTuner(P,'PI');
Kp_pt = 313.7;
Ki_pt = 44.5;
C_pt = pid(Kp_pt,Ki_pt);
W_pt = feedback(C_pt*P,1)
```

```
W_pt =

      6.274 s + 0.89
  -----------------------
  20 s^2 + 7.274 s + 0.89

Continuous-time transfer function.
Model Properties
```

```
% Risposta al gradino dei due sistemi.
u = 10; % Ampiezza del gradino [m/s].
respCfg = RespConfig('Amplitude',u); % Configurazione del gradino.

% FdT con PI tarato manualmente.
step(W,respCfg);
hold on;

% FdT con PI tarato con pidTuner.
step(W_pt,respCfg)
grid on;

% Plot linee errore a regime: 2 [%].
plot([0, 35], [10.2, 10.2],'--','Color','magenta');
hold on
plot([0, 35], [9.8, 9.8],'--','Color','magenta');
hold on

% Plot linee tratteggiate massima sovraelongazione: 10 [%].
plot([0, 11], [11, 11],'--','Color','g');
hold on;
plot([11, 11], [0, 11],'--','Color','g');
hold on

% Plot linea tratteggiata tempo di assestamento: 23.8 [s].
plot([23.8, 23.8], [0, 10.2],'--','Color','red');
grid on;
hold off;

% Legenda.
title('Risposta al gradino');
xlabel('Tempo [s]');
ylabel('Velocità veicolo [m/s]');
legend('PI: sintesi manuale','PI: sintesi pidTuner', ...
       'Errore a regime: +2 [%]', 'Errore a regime: -2 [%]', ...
       'Massima sovrael: 10 [%]','Tempo [s]','Tempo di assestamento [s]');
legend('Location','southeast');
```



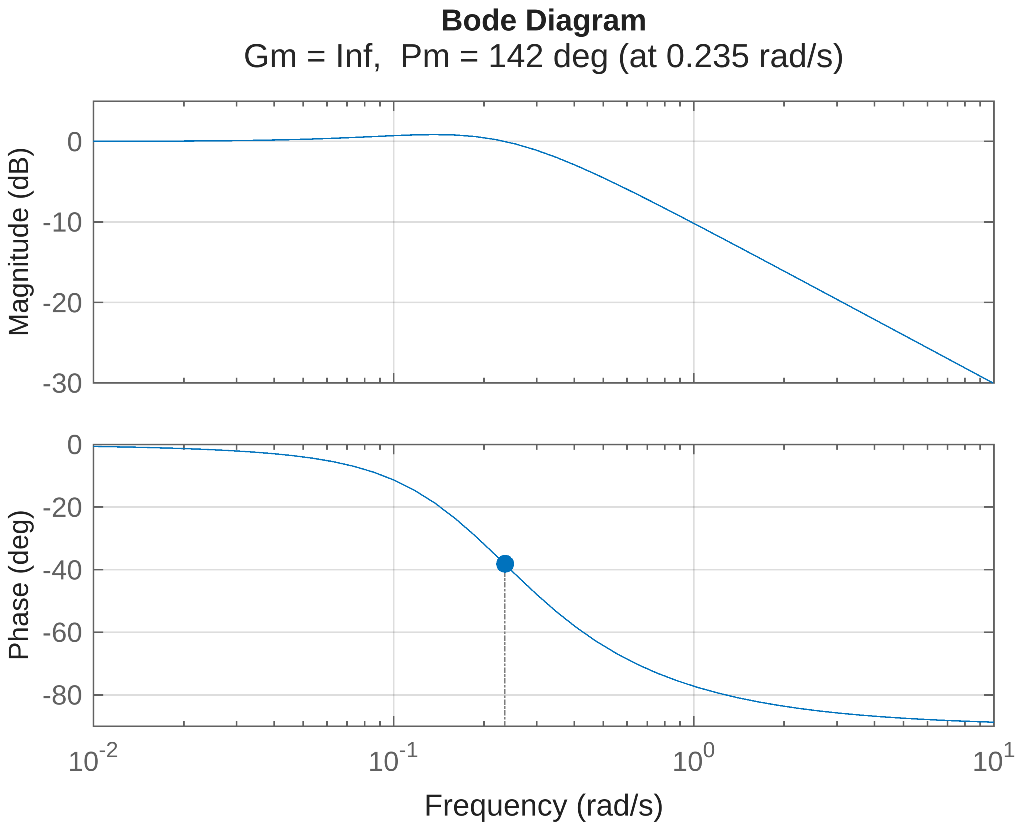
Stabilità del sistema.

Come si può vedere dal grafico della risposta a gradino, la linea arancio (taratura PI con pidTuner) presenta due poli complessi coniugati. Questo significa che il **sistema è stabile** e ha un transitorio rapido con accumulo e dissipazione oscillante dell'energia. La stabilità è anche confermata dal fatto che i 2 poli sono a parte reale negativa e dall'ampio margine di fase (dagramma di Bode).

```
% Poli.
W_pt_poli = pole(W_pt)
```

```
W_pt_poli = 2×1 complex
  -0.1819 + 0.1069i
  -0.1819 - 0.1069i
```

```
% Margine di fase.
margin(W_pt);
grid on;
```



Verifica delle specifiche di prestazione.

Tramite la seguente formula si verifica se la funzione di trasferimento calcolata tramite **Matlab pidTuner** soddisfa le specifiche di progetto. Si calcola il **coefficiente di smorzamento** ξ che determina il massimo valore di **sovraelongazione** M_p (10%) ed il valore di α che indica il valore minimo della parte reale dei poli complessi coniugati per ottenere un **tempo di assestamento** T_s al 2%.

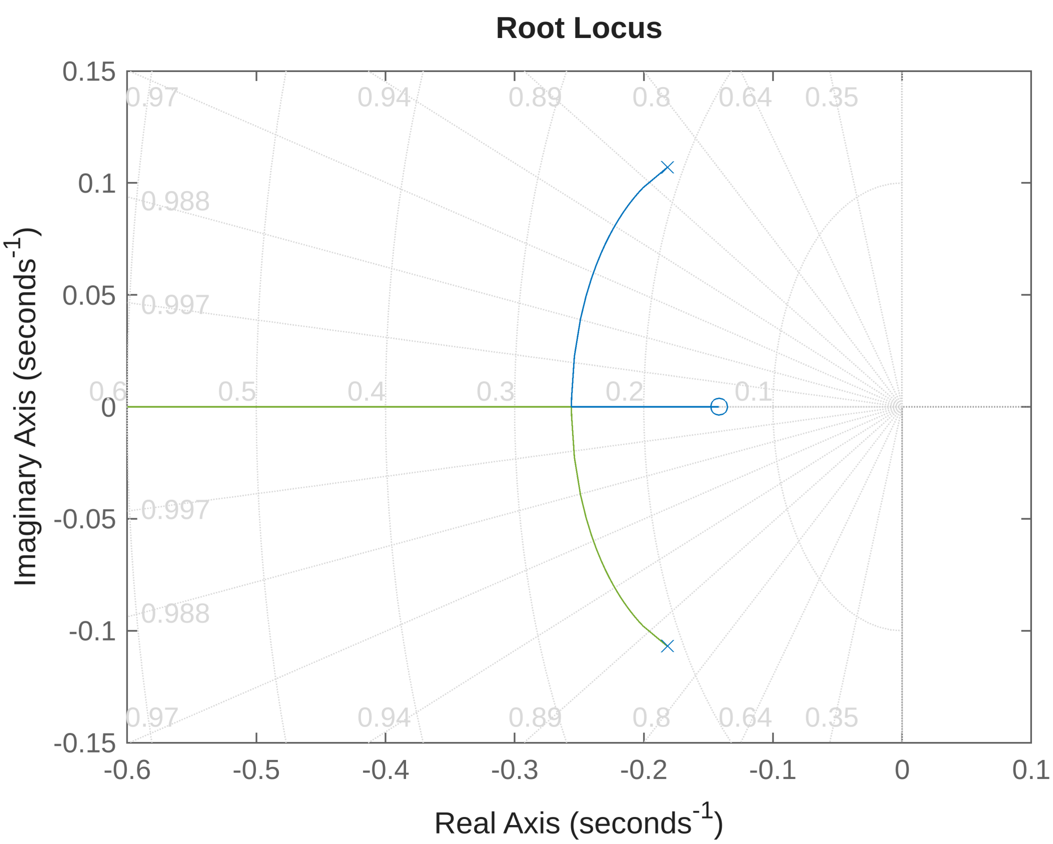
$$\xi \geq \sqrt{\frac{\ln^2(M_p)}{\pi^2 + \ln^2(M_p)}} = \sqrt{\frac{\ln^2(10)}{\pi^2 + \ln^2(10)}} \approx 0.6$$

Per ottenere un tempo di assestamento entro il 2% si deve risolvere la seguente disuguaglianza:

$$e^{-\zeta\omega_n T_s} \leq e_{\mathrm{rr}\%} \rightarrow T_s \geq \frac{\ln(e_{\mathrm{rr}\%})}{\zeta\omega_n} = \frac{\ln(0.02)}{0.6 \cdot 0.235} \approx 28\ [s]$$

Dal grafico della risposta a gradino si può notare si che il valore massimo di **sovraelongazione** è al 10% ed il **tempo di assestamento** a $23.8[s]$.

```
Psi = 0.6; % Coefficiente di Smorzamento [-].
Wn = 0.235; % Pulsazione Naturale sistema ad anello chiuso [rad/s].
a = Psi*Wn;

% Plot del Luogo delle Radici.
rlocus(W_pt)
sgrid(Psi,a)
```



Dal diagramma del **Luogo delle Radici** si nota che i **poli** sono negativi (al variare di K da 0 a ∞) e quindi il sistema è stabile e che i 2 poli complessi coniugati sono :

- $> \zeta$: garantisce sovraelongazione.
- $> \alpha$: garantisce tempo di assestamento.

Errore a regime (Teorema del valore finale).

Il **teorema del valore finale** permette di determinare il comportamento a regime (cioè per t che tende ad infinito o s che tende a 0) di un sistema dinamico lineare tempo-invariante (LTI) a partire dalla sua funzione di trasferimento nel dominio di Laplace.

$$\lim_{t \to \infty} x(t) = \lim_{s \to 0} s \cdot X(s)$$

Il sistema in questione risulta essere **stabile** e quindi è possibile applicare il teorema:

$$Ess = \lim_{s \to 0} s \cdot E(s) = s \frac{U(s)}{1 + L(s)} = s \cdot \frac{1}{s} \cdot \frac{1}{\frac{K_i}{b} \cdot \frac{1}{s}} = \frac{1}{\infty} \approx 0$$

dove si ha:

- $E(s)$: errore finale.
- $U(s)$: segnale in ingresso al sistema (segnale a gradino in questo caso).
- $L(s)$: funzione ad anello aperto.

Il sistema risulta avere valore **nullo** a regime.