Fondamenti di Automatica

Introduzione a Matlab
(con Symbolic Toolbox e Control Systems Toolbox)

Prof. Marcello Bonfè

Dipartimento di Ingegneria - Università di Ferrara

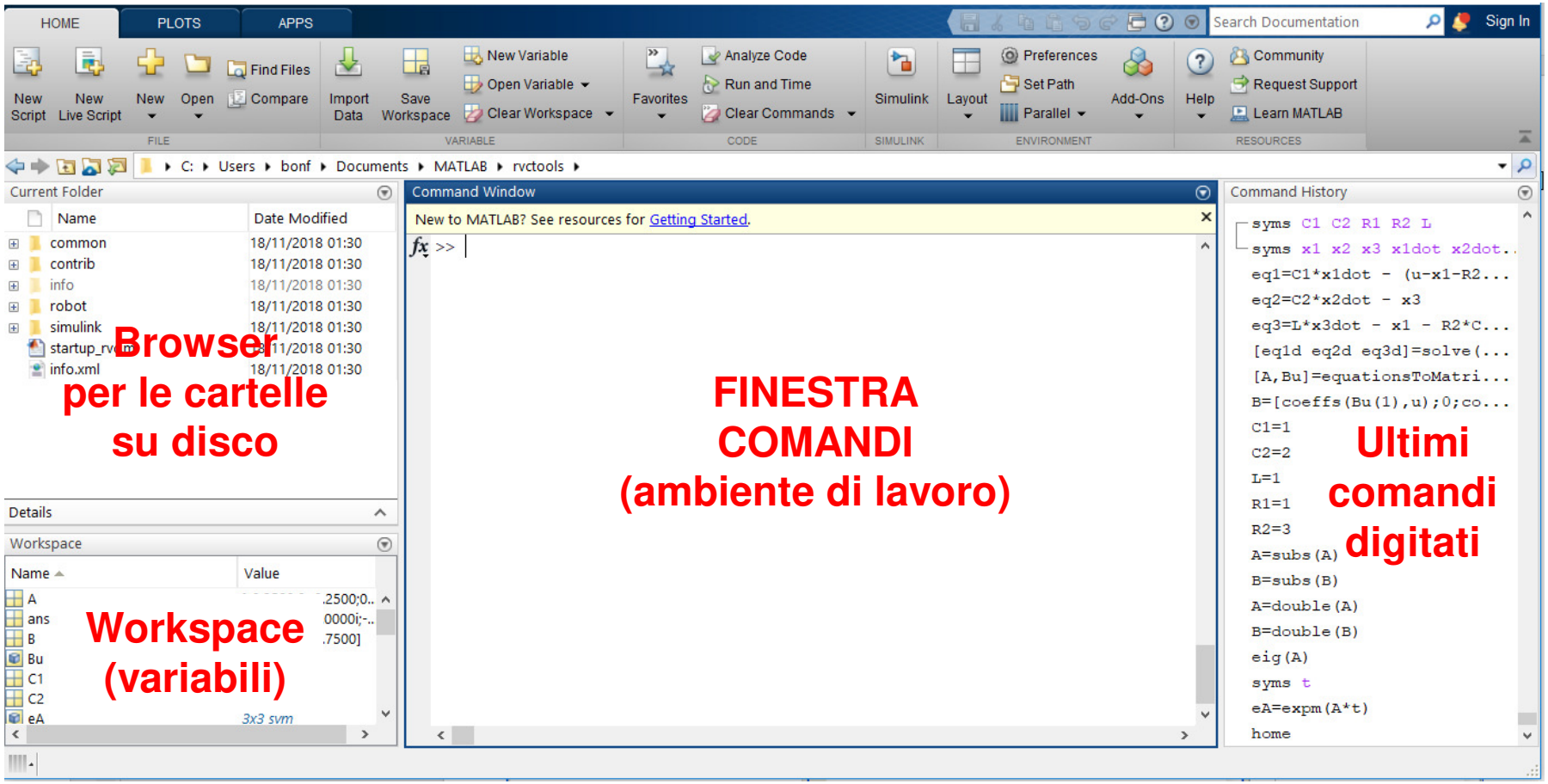
Tel. +39 0532 974839

E-mail: marcello.bonfe@unife.it

DE: Department of Engineering Ferrara

Università degli Studi di Ferrara

Matlab: interfaccia principale



NOTA BENE: Symbolic Toolbox e Control System Toolbox NON sono installati nell'installazione di default, occorre quindi selezionarli in modo specifico facendo l'installazione personalizzata, oppure installarli in seguito, dal tab *Apps* (poi *Get more Apps* e ricerca per nome)

Matlab: definizione di variabili, vettori e matrici

Definire variabile scalare

```
>> x = 3
```

Definire vettore riga (1×3)

```
>> x = [1 2 3]
```

Idem, ma senza echo dell'output

```
>> x = [1 2 3];
```

Definire vettore colonna (3x1)

```
>> x = [1; 2; 3]
```

(oppure `>> x = [1 2 3]'`)

Definire matrice 3x4

```
>> A = [1 2 3 4;5 6 7 8;9 10 11 12]
```

Accedere / modificare elemento di riga 2 e colonna 1

```
>> A(2,1) = 0
```

Matlab: operazioni su matrici

- Le "solite" operazioni matematiche: **`+,-,*,/,^`**
- **Es. `>> A^3`** (potenza di matrice, solo se quadrata!)
- Precedute dal punto, sono eseguite elemento per elemento anziché in senso matriciale/vettoriale
- Operazioni specifiche per matrici / vettori:
  - Trasposta: **`A'`**
  - Determinante: **`det(A)`**
  - Inversa: **`inv(A)`**
  - Autovalori: **`eig(A)`**
  - Rango: **`rank(A)`**
  - Polinomio caratteristico: **`poly(A)`**
  - Esponenziale di matrice: **`expm(A)`**
  - Radici di un polinomio: **`roots(x)`** (x vettore dei coeff.)

Matlab: inizializzazione di matrici *standard*

- Comandi che forniscono matrici caratteristiche, utili per inizializzare variabili opportune:
  - Matrice m x n con tutti elementi nulli: **`zeros(m,n)`**

    NOTA: **`zeros(m)`** fornisce matrice quadrata
  - Matrice m x n con tutti elementi unitari: **`ones(m,n)`**

    NOTA: **`ones(m)`** fornisce matrice quadrata
  - Identità n x n: **`eye(n)`**
  - Matrice quadrata diagonale (con elementi sulla diagonale nel vettore V): **`diag(V)`**

Matlab: il workspace

- I risultati di tutti i comandi digitati vengono memorizzati nel cosiddetto workspace della sessione
- Il workspace viene cancellato all'uscita dal Matlab!
- Il contenuto del workspace si può salvare (anche parzialmente) e ripristinare:
  - **`save nomefile`** (estensione di default: **.mat**)
  - **`save nomefile variabile1 variabile2`** (salva solo le variabili indicate)
  - **`load nomefile`**
  - **`clear`**: cancella il contenuto del workspace!!

Matlab: soluzione di sistemi (calcolo simbolico)

- La funzione **`solve(eqns,vars)`** risolve il sistema di equazioni **`eqns`** nelle variabili specificate **`vars`**

```
>> syms x1 x2 c1 c2 c3 x1dot x2dot u;
>> eqns = [x1dot + x2 + c1*u == 0; c2*(x1 + x2dot) == c3*u];
>> vars = [x1dot;x2dot]
>> sol = solve(eqns,vars)

sol =

  struct with fields:
    x1dot: [1×1 sym]
    x2dot: [1×1 sym]
```

Matlab: soluzione di sistemi (calcolo simbolico)

- La funzione **`solve(eqns,vars)`** risolve il sistema di equazioni **`eqns`** nelle variabili specificate **`vars`**

```
>> sol.x1dot
ans =

-x2 - c1*u
```

```
>> sol.x2dot
ans =

(c3*u - c2*x1)/c2
```

NOTA: per accedere ai campi di una struttura si utilizza l'operatore «`.`».
In alternativa si può utilizzare l'assegnamento diretto:
`[x1dot,x2dot] = solve(eqns,vars)`

Matlab: soluzione di sistemi (calcolo simbolico)

- Con l'esempio visto si è ottenuta di fatto l'espressione di equazioni differenziali accoppiate predisposta per la scrittura del modello di un sistema dinamico nello spazio degli stati:

$$\begin{aligned} \dot{x}_1 + x_2 c_1 u &= 0 \\ c_2(x_1 + \dot{x}_2) &= c_3 u \end{aligned} \quad \Rightarrow \quad \begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases}$$

- Cerchiamo ora un metodo furbo per estrarre i coefficienti delle matrici…

Matlab: operazioni su polinomi

- La funzione **`collect(expr,x)`** riorganizza i termini del polinomio **`expr`** in modo che siano raccolti i coefficienti delle potenze della variabile **`x`**

```
>> syms x;
>> expr = 3*x^2 + 23*(2*x-x^4)*(x-k)+x^5;
>> collect(expr,x)
ans =
    - 22*x^5 + 23*k*x^4 + 49*x^2 - 46*k*x
```

Matlab: operazioni su polinomi (calcolo simbolico)

- La funzione **`coeffs(expr,x)`** restituisce i coefficienti del polinomio **`expr`** rispetto alla variabile simbolica **`x`**

```
>> syms x;
>> expr = 3*x^2 + 23*(2*x-x^4)*(x-k)+x^5;
>> coeffs(expr,x)
ans =
  [ -46*k, 49, 23*k, -22]
```

NOTA: il vettore restituito NON contiene i coefficienti NULLI ed è ordinato a partire dal coefficiente associato al termine con potenza minore.

Matlab: sistemi in forma matriciale

- La funzione **`[A,B]=equationsToMatrix(eqns,x)`** restituisce la matrice **`A`** e il vettore dei termini noti **`B`** del sistema di equazioni **`eqns`** tali che **`A*x = B`**

```
>> syms x1 x2 u;
>> eqns = [x1+21*x2 == -3*u;-2*x1-x2==0];
>> x = [x1;x2];
>> [A,Bu] = equationsToMatrix(eqns,x)

A =                    Bu =
[  1, 21]               -3*u
[ -2, -1]                0
```

Matlab: sistemi in forma matriciale

- **NOTA:** il sistema considerato è nella forma A*x=Bu. Pertanto, il secondo risultato fornito, indicato come vettore Bu nell'esempio precedente:
  - **Non** rappresenta i coefficienti della matrice B nel generico modello nello spazio degli stati:

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

    ma rappresenta il prodotto B*u (detta *azione forzante*)
  - E' a secondo membro, quindi va cambiato di segno per ricavarne i coefficienti della matrice B cercata:

```
>> B=-Bu/u
B =
     3
     0
```

← SOLO se u è scalare!

```
>> B=[coeffs(Bu(1),u);
      coeffs(Bu(2),u)]
```

ANCHE se u è un vettore
(ma attenzione ai coefficienti nulli..)

Matlab: sistemi in forma matriciale

- **RIASSUMENDO:** la combinazione di comandi più efficace per estrarre in pochi passi le matrici A e B del modello nello spazio degli stati di un sistema dinamico è:

1. Scrivere le equazioni in forma simbolica, includendo simboli per le derivate degli stati (es. **`x1dot, x2dot,`** ecc)
2. Risolvere le equazioni rispetto alle derivate degli stati (i.e. **`solve(eqns,[x1dot x2dot …])`**)
3. Eseguire la equationsToMatrix sulle espressioni ottenute per le derivate degli stati, rispetto alle variabili di stato:
   `[A,Bu]=equationsToMatrix([x1dot; x2dot ..],[x1;x2;..)`
4. Scorporare l'ingresso dalla matrice Bu ottenuta, cambiandone il segno (es. caso tipico di sistema con ingresso scalare: **`B= -Bu/u`**)

Matlab: analisi del sistema dinamico ottenuto

- Si è mostrato in aula che la soluzione dell'equazione differenziale matriciale che descrive il sistema dinamico nello spazio degli stati:

$$\dot{x}(t) = Ax(t); \quad x(0) = x_0,\ x(t) \in \mathbb{R}^n$$

richiede il calcolo dell'esponenziale di A:

$$x(t) = e^{At}x_0$$

Matlab: analisi del sistema dinamico ottenuto

- La matrice e^{At} si può calcolare manualmente con un procedimento mostrato in aula detto *metodo del polinomio interpolante*
- Punto di partenza del metodo: calcolo degli autovalori di A
- In Matlab, con A sia numerica che simbolica:

```
>> eig(A)
```

- Ricordiamo che gli autovalori sono le radici del polinomio caratteristico ottenuto risolvendo

$$\det(\lambda I - A) = 0$$

Matlab: analisi del sistema dinamico ottenuto

- In Matlab:

```
>> syms lambda
>> polycar = det(lambda*eye(2) - A)
>> eigenvals = solve(polycar == 0)
eigenvals =

 -41^(1/2)*1i
  41^(1/2)*1i
```

Matlab: esponenziale di matrice (calcolo simbolico)

- In Matlab, è necessario definire la matrice A e il simbolo `t`:

```
>> A=[-4 0; 1 -4]
>> syms t
>> expm(A*t)
ans =
[ 1/exp(4*t),           0]
[ t/exp(4*t), 1/exp(4*t)]
```

NOTA: il risultato è simbolico, i termini esponenziali sono a denominatore, il che equivale ad esponente negativo

Matlab: esponenziale di matrice (calcolo simbolico)

- Nota la matrice esponenziale, è possibile calcolare il valore dello stato di un sistema dinamico noto lo stato iniziale e il tempo intercorso tra i due stati

```
>> x3=[1; 0]
>> x4=expm(A*(4-3))*x3
x4 =
    0.0183
    0.0183
```

NOTA: il risultato numerico equivale a e^{-4} (in Matlab `exp(-4)`) per entrambe le variabili di stato..

Matlab: Control System Toolbox

- La funzione **`sys=ss(A,B,C,D)`** crea l'oggetto rappresentativo del modello nello spazio degli stati a partire dalle matrici A,B,C,D

```
>> sys = ss([-2 0;-1 -11],[1;0],[0 1],2)

sys =

  A =
        x1   x2
   x1   -2    0
   x2   -1  -11

  B =
       u1
   x1   1
   x2   0

  C =
        x1  x2
   y1    0   1

  D =
        u1
   y1    2

Continuous-time state-space
model.
```

Matlab: Control System Toolbox

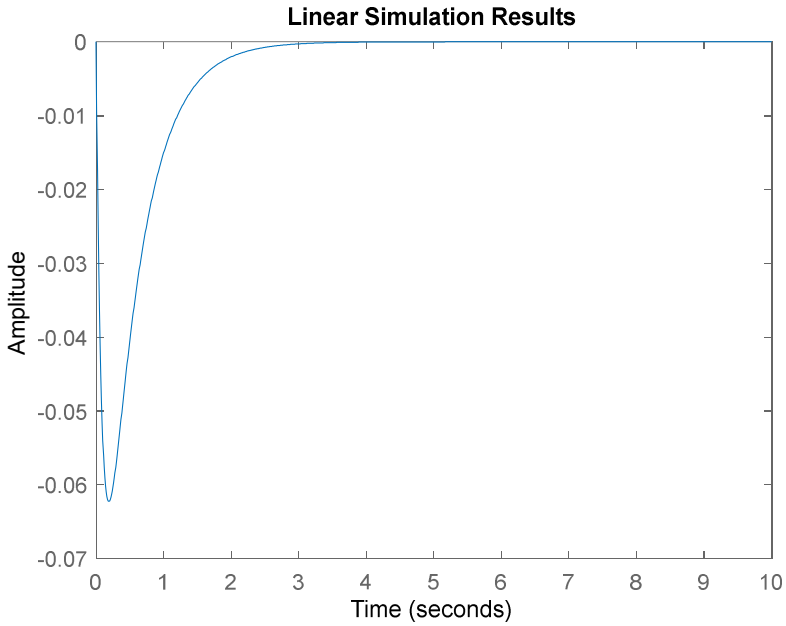
- La funzione **`y=lsim(sys,u,t,x0)`** simula l'andamento nel tempo del sistema a partire dalle condizioni iniziali e ne restituisce l'uscita

```
>> t=[0:0.01:10];
>> u=zeros(size(t));
>> x0 = [1 0];
>> y=lsim(sys,u,t,x0);
```

NOTA: se chiamata così

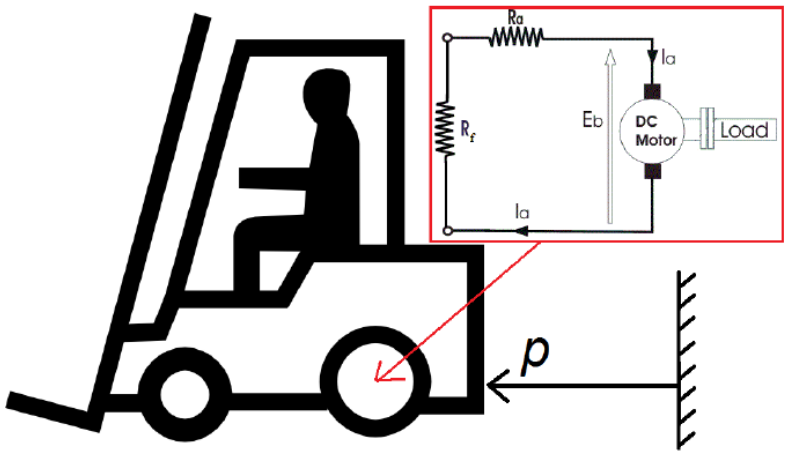
```
>> lsim(sys,u,t,x0)
```

viene aperta la finestra di analisi grafica per sistemi LTI → → → → →



Esercizio 1

- Modello di un carrello elevatore a trazione elettrica in modalità di frenatura



$$m\ddot{p} + b\dot{p} + \frac{k_m^2}{R_a + R_f}\dot{p} = 0$$

$$x_1 = p;\ x_2 = \dot{p};$$

m = 1000; b = 100; R_a = 10; R_f = 90 ; k_m = 300;

Si determini lo spazio percorso e la velocità raggiunta in 12 secondi dal veicolo (i.e. x(t) con t=12) in modalità di frenata, considerando una velocità iniziale di 5m/s, vale a dire:

$$x(0) = [0 \quad 5]^T$$

Soluzione esercizio 1

- Ricavare il sistema di equazioni esplicitando i termini derivativi

```
>> syms m km Ra Rf b
>> syms x1 x2 x1dot x2dot u y t

>> x = [x1;x2];
>> xdot = [x1dot;x2dot];
>> eqns = [x1dot == x2; m*x2dot+ b*x2 + (km^2/(Ra +
Rf))*x2==0];

>> [x1dot,x2dot] = solve(eqns,xdot)
x1dot =
x2

x2dot =
-(x2*(km^2 + Ra*b + Rf*b))/(m*(Ra + Rf))
```

Soluzione esercizio 1

- Ricavare le matrici A,B

```
>> [A,Bu]=equationsToMatrix([x1dot;x2dot],x)
A =
[ 0,                                        1]
[ 0, -(km^2 + Ra*b + Rf*b)/(m*(Ra + Rf))]

Bu =
 0
 0

>> B=-Bu/u

B =

 0
 0
```

Soluzione esercizio 1

- Ricavare la matrice A in forma numerica

```
>> m = 1000;
>> b = 100;
>> Ra = 10;
>> Rf = 90;
>> km = 300;

>> A=subs(A);
>> A=double(A)
A =

     0     1
     0    -1
```

- Ricavare l'esponenziale di matrice in forma simbolica

```
>> eAt = expm(A*t)

eAt =

[ 1, 1 - exp(-t)]
[ 0,     exp(-t)]
```

Soluzione esercizio 1

- Risolvere l'esercizio in forma numerica

```
>> x0 = [0;5];
>> tf = 12;

x_12 = expm(A*tf)*x0

x_12 =

    5.0000
    0.0000
```

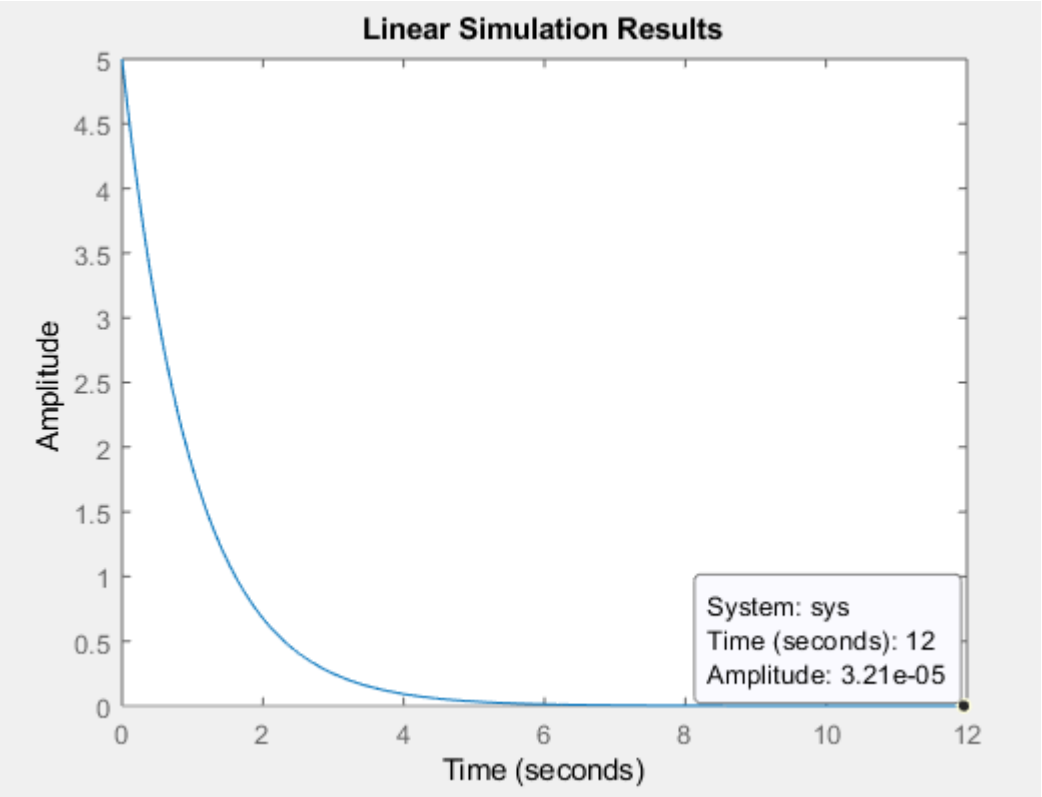
NOTA: il risultato è arrotondato, di default Matlab mostra solo 4 cifre dopo il punto decimale. Per mostrare più cifre:

```
>> format long
>> x_12
x_12 =
   4.999969278938233
   0.000030721061767
```

Soluzione esercizio 1

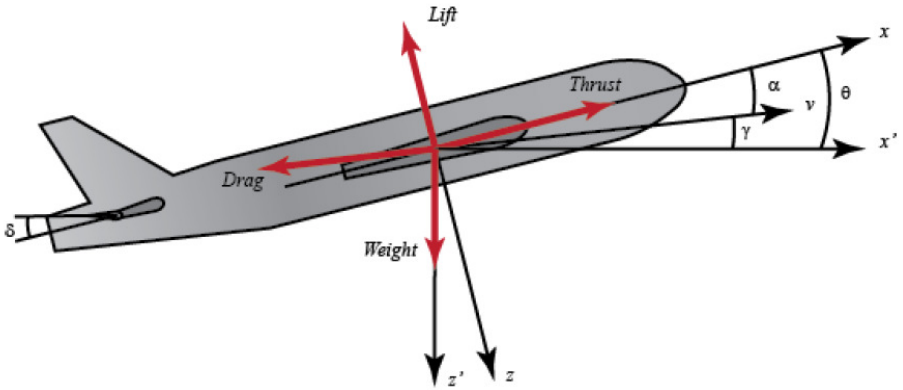
- Confronto con lsim nel caso $y = x_2$

```
>> C = [0 1];
>> D = 0;
>> sys = ss(A,B,C,D);
>> t=[0:0.01:12];
>> u=zeros(size(t));
>> x0 = [0 5];
>> lsim(sys,u,t,x0)
```



Esercizio 2

- Modello semplificato della dinamica longitudinale di un aereo



[Fonte: Control Tutorials for Matlab&Simulink, http://ctms.engin.umich.edu]

$$\dot{\alpha} = -2\alpha + C_q q + 2\delta$$
$$\dot{q} = -\alpha - 4q + \delta$$
$$\dot{\theta} = C_q q$$

$$x_1 = \alpha;\ x_2 = q;\ x_3 = \theta;\ u = \delta;\ y = \theta;$$

Si determini l'uscita all'istante t=10 data la condizione iniziale x0 = [1;0;0], t0=0 con Cq=1

Soluzione esercizio 2

```
>> syms x1 x2 x3 x1dot x2dot x3dot u Cq

>> eqns = [x1dot == -2*x1+Cq*x2+2*u;x2dot==-x1-
4*x2+u;x3dot==Cq*x2];
>> [x1dot,x2dot,x3dot]=solve(eqns,[x1dot;x2dot;x3dot])
>> [A,Bu]=equationsToMatrix([x1dot;x2dot;x3dot],[x1;x2;x3])
>> B=-Bu/u;
% oppure
>> B=-[coeffs(Bu(1),u);coeffs(Bu(2),u);0];
>> C=[0 0 1];
>> D=0;
>> Cq=1;
>> A=double(subs(A));
>> x0=[1;0;0];
>> tf=10;
>> xf=expm(A*tf)*x0;
>> yf=C*xf
```

INTRODUZIONE A MATLAB

FINE