Fondamenti di Automatica

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

Prof. Marcello Bonfè

Dipartimento di Ingegneria - Università di Ferrara

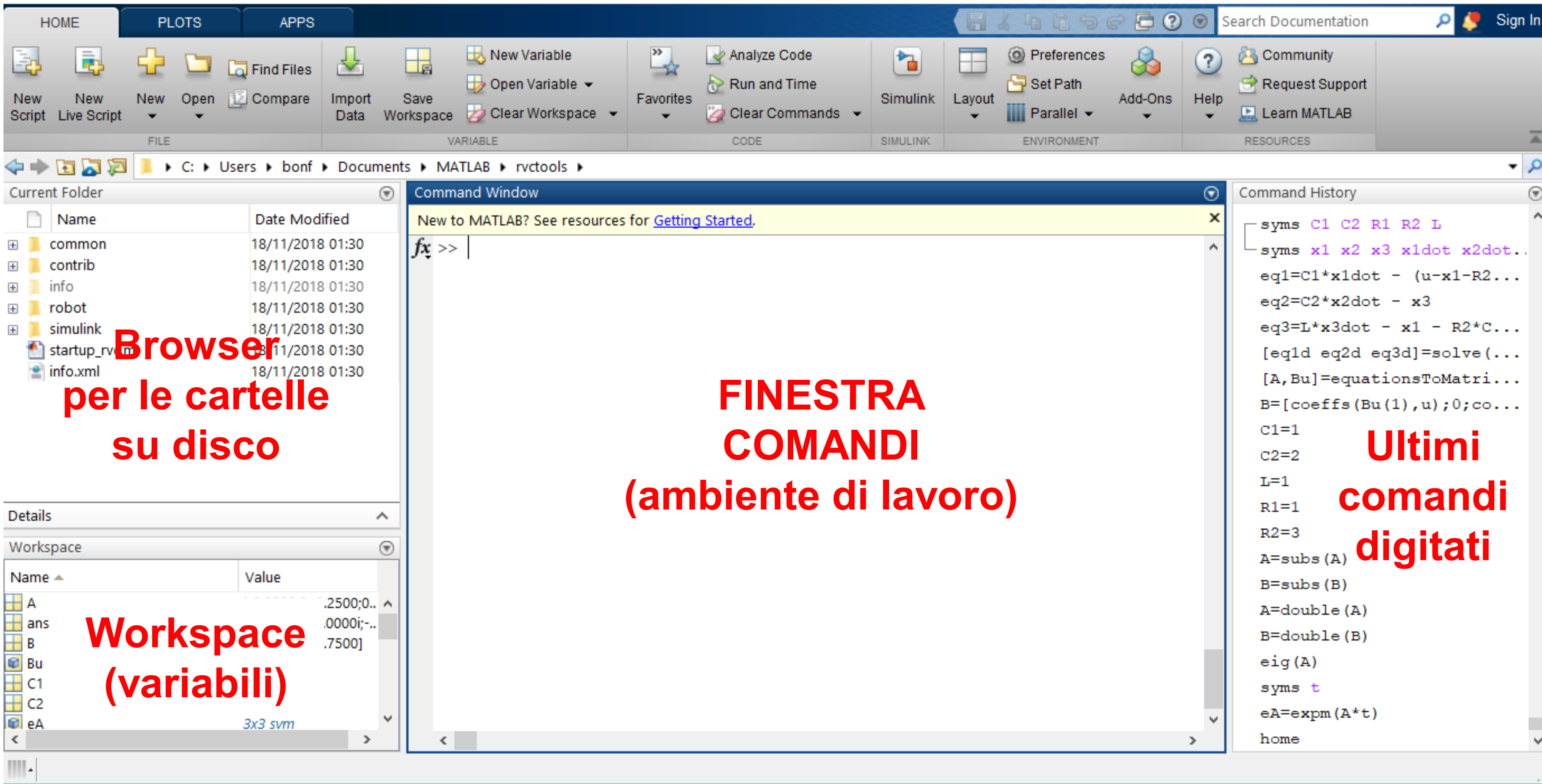
Tel. +39 0532 974839

E-mail: marcello.bonfe@unife.it

Department of Engineering Ferrara

Università degli Studi di Ferrara

Matlab: interfaccia principale



NOTA BENE: Symbolic Toolbox e Control System Toolbox possono essere installati assieme a Matlab durante la procedura di installazione iniziale personalizzata, oppure installati in seguito aprendo Matlab ed selezionando il menu *Apps* → *Get more Apps* e ricercandoli 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` (<u>potenza di matrice</u>, solo se quadrata!)

NOTA: precedute dal punto (es. `A.*B, A.^3`) sono svolte <u>elemento per elemento</u> 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

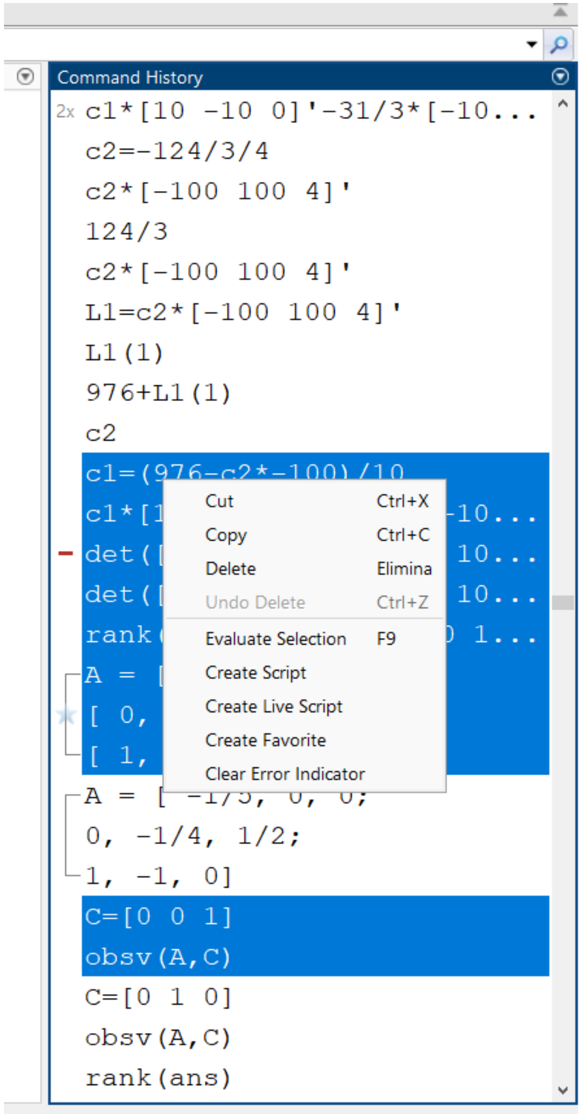
- I risultati di tutti i comandi digitati vengono memorizzati nel cosiddetto workspace della sessione
- Il workspace viene cancellato all'uscita dal Matlab!
- Il contenuto del workspace si può salvare (anche parzialmente) e ripristinare:
  - **`save nomefile`** (estensione di default: **.mat**)
  - **`save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

Matlab: gli script file

- Per automatizzare l'esecuzione di una sequenza di comandi è possibile creare degli script file
- Uno script file è un file di testo, salvato con estensione **`.m`**, che può essere poi eseguito digitandone il nome dalla *Command Window*, purchè il file sia nella *Current Folder* (oppure in una cartella memorizzata nel ***Path*** di Matlab)
- Oltre all'ovvio passaggio dal menu "New Script" (tab "Home"), è utile ricordare la possibilità di copiare direttamente in un nuovo file i comandi eseguiti in passato dalla finestra *Command History*

Matlab: gli script file

- Dalla finestra *Command History*
  - selezionare le righe di interesse con il <u>tasto sinistro</u> del mouse + <u>CTRL</u>
  - cliccare su una delle righe selezionate con il <u>tasto destro</u> del mouse
  - cliccare “create script”



Strumenti per la soluzione degli esercizi sulla modellazione di sistemi ingegneristici

Matlab: soluzione di sistemi (calcolo simbolico)

- La funzione **`solve(eqns,vars)`** risolve il sistema di equazioni **`eqns`** nelle variabili simboliche (dichiarate con **`syms`**) **`vars`**

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

x1dot =
- x2 - c1*u


x2dot =
(c3*u - c2*x1)/c2
```

Matlab: soluzione di sistemi (calcolo simbolico)

- **NOTA1**: le variabili `x1,x2,` ecc. potrebbero comparire inizialmente con un simbolo differente, per sostituire il quale si può usare il comando `subs(s,old,new)`
- **NOTA2**: le variabili di stato sono **funzioni del tempo**, le cui derivate andrebbero indicate con `diff(x1(t),t)`, ma in questo contesto l'obiettivo è rielaborare le equazioni in modo algebrico, <u>NON risolverle come equazioni differenziali</u>, perciò sono indicate come **pura notazione**

```
>> syms Vc Ps c1 c2 c3 dotVc dotPs u;
>> eqns = [dotVc + Ps + c1*u == 0; c2*(Vc + dotPs) == c3*u];
>> syms x1 x2 x1dot x2dot;
>> eqns = subs(eqns,[Vc Ps dotVc dotPs],[x1 x2 x1dot x2dot])
>> xdot = [x1dot,x2dot];
...
```

Matlab: soluzione di sistemi (calcolo simbolico)

- **NOTA3**: se nelle equazioni compaiono variabili ausiliarie rispetto a stati e ingresso, si possono rimpiazzare automaticamente con **`eliminate(eqns,vars)`**

```
>> syms Vc Ps Pq c1 c2 c3 dotVc dotPs u;
>> eqns = [dotVc+Ps+c1*u==0; c2*Pq==c3*u; Pq==Vc+dotPs];
>> eqns = eliminate(eqns,Pq)
>> syms x1 x2 x1dot x2dot;
>> eqns = subs(eqns,[Vc Ps dotVc dotPs],[x1 x2 x1dot x2dot])
>> xdot = [x1dot,x2dot];
...
```

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: 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`**

```
>> xdot = [x1dot,x2dot]
```

← **NOTA:** comando da eseguire <u>nuovamente</u> **DOPO** aver calcolato la soluzione simbolica di **`x1dot, x2dot ecc.`**

```
>> x = [x1,x2]
>> [A,Bu] = equationsToMatrix(xdot,x)

      A =
      [  0, -1]
      [ -1,  0]
      Bu =
             c1*u
      -(c3*u)/c2
```

Matlab: sistemi in forma matriciale

- **NOTA:** i risultati finali della slide precedente sono in 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*)
  - È a secondo membro, quindi va cambiato di segno per ricavarne i coefficienti della matrice B cercata:

```
>> B=-Bu/u
```
←**SOLO** se u è scalare (**sistemi Single-Input**..)!

```
    B =
           -c1
         c3/c2
```

Matlab: sistemi in forma matriciale

- **RIASSUMENDO,** per estrarre le matrici A e B:

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

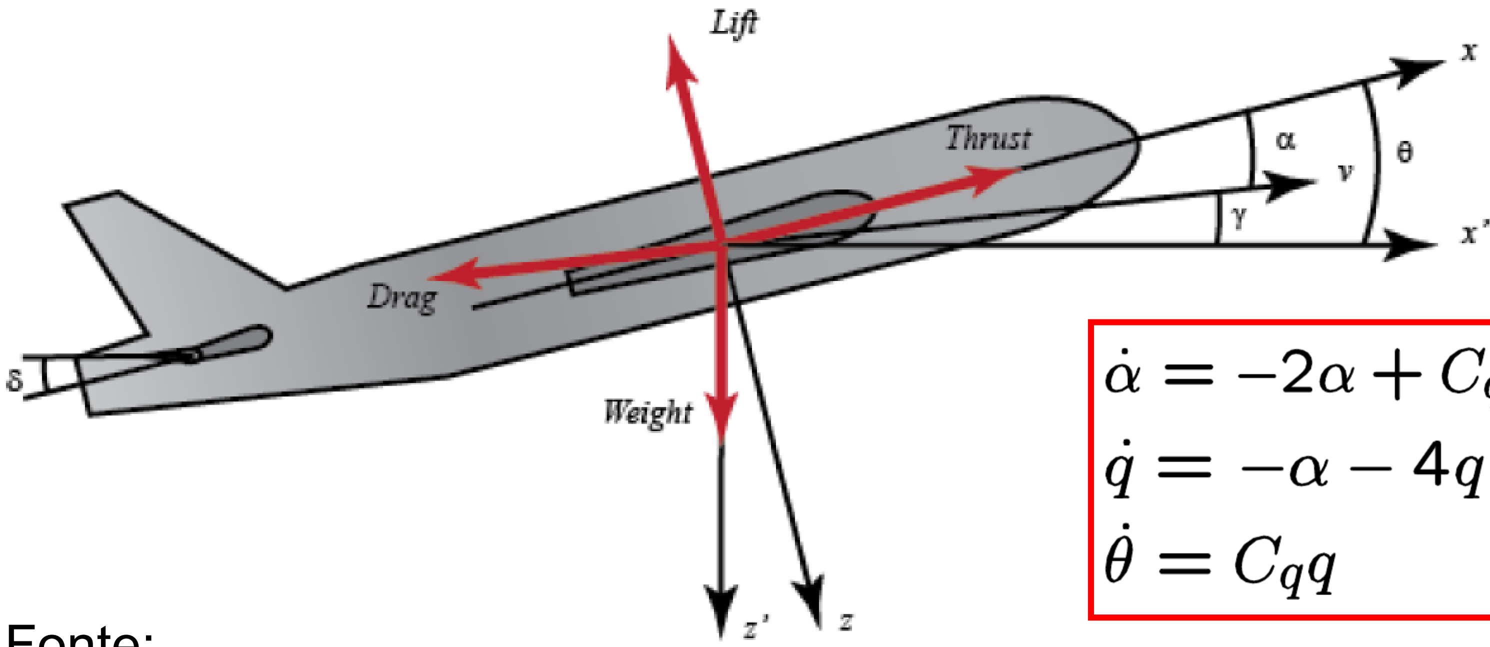
Matlab: sistemi in forma matriciale

- Per ricavare le matrici C e D valgono considerazioni analoghe, ricordando però che queste compaiono nell'equazione algebrica y=C*x+D*u e che, pertanto, NON devono contenere variabili che siano rappresentative di derivate rispetto al tempo
- Qualora tali derivate siano presenti, vanno sostituite con opportune espressioni (es. comando **`eliminate()`**)

```
>> y = x2
>> x = [x1,x2]
>> [C,Du] = equationsToMatrix(y,x)
   C = [0, 1]
   Du = 0
```

Esercizio riepilogativo:

- Modello semplificato della dinamica longitudinale di un aereo



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

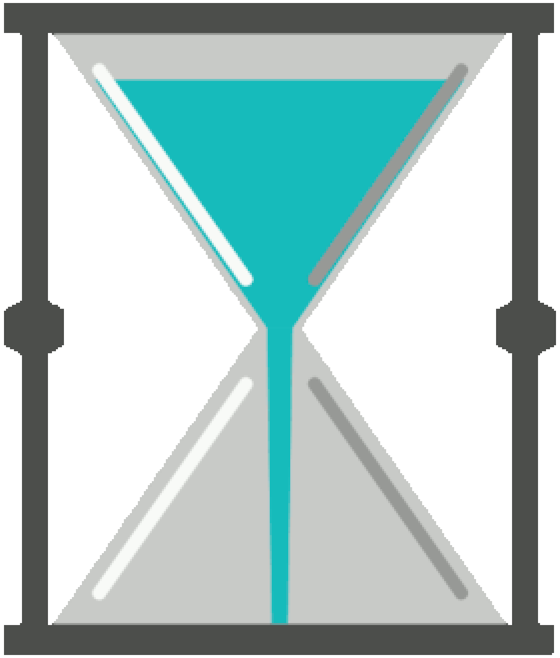
Fonte:
https://ctms.engin.umich.edu/

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

Si determinino le matrici A,B,C,D del modello: $\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases}$

Esercizio riepilogativo:

Let’s do it!!



Strumenti per la soluzione degli esercizi sull’analisi di sistemi: transizione dello stato

Matlab: analisi del sistema dinamico

- 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*t:

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

Matlab: analisi del sistema dinamico

- 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: matrice esponenziale di A*t (simbolica)

- In Matlab, il calcolo è eseguito con il comando **`expm()`**, definita 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)]
```

NOTA1: il risultato è simbolico, i termini esponenziali sono a denominatore, il che equivale ad esponente negativo.

NOTA2: il comando **`exp()`** eseguito sulla matrice A*t è <u>elemento per elemento</u>, quindi **NON** è equivalente a **`expm()`**!!!

Matlab: matrice esponenziale di A*t (simbolica)

- 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]
```

$$x(3) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

```
>> x4=expm(A*(4-3))*x3
```

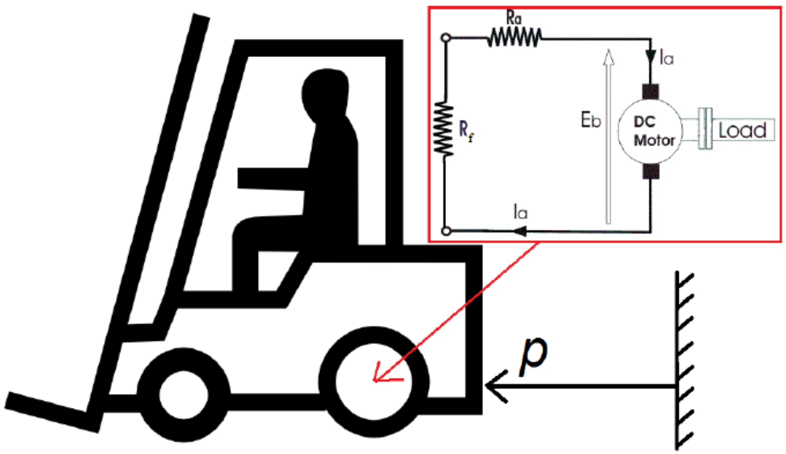
$$x(4) = e^{A(4-3)}x(3)$$

```
x4 =

    0.0183

    0.0183
```

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

Esercizi 1 e 2 – Prova "Tipo" B

- 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

>> 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

```
>> xdot = [x1dot;x2dot];
```
DA RIPETERE, DOPO la `solve()`!!

```
>> [A,Bu]=equationsToMatrix(xdot,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 2

- 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 2

- Risolvere l'esercizio in forma numerica

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

x_12 = expm(A*tf)*x0

x_12 =

    5.0000
    0.0000
```

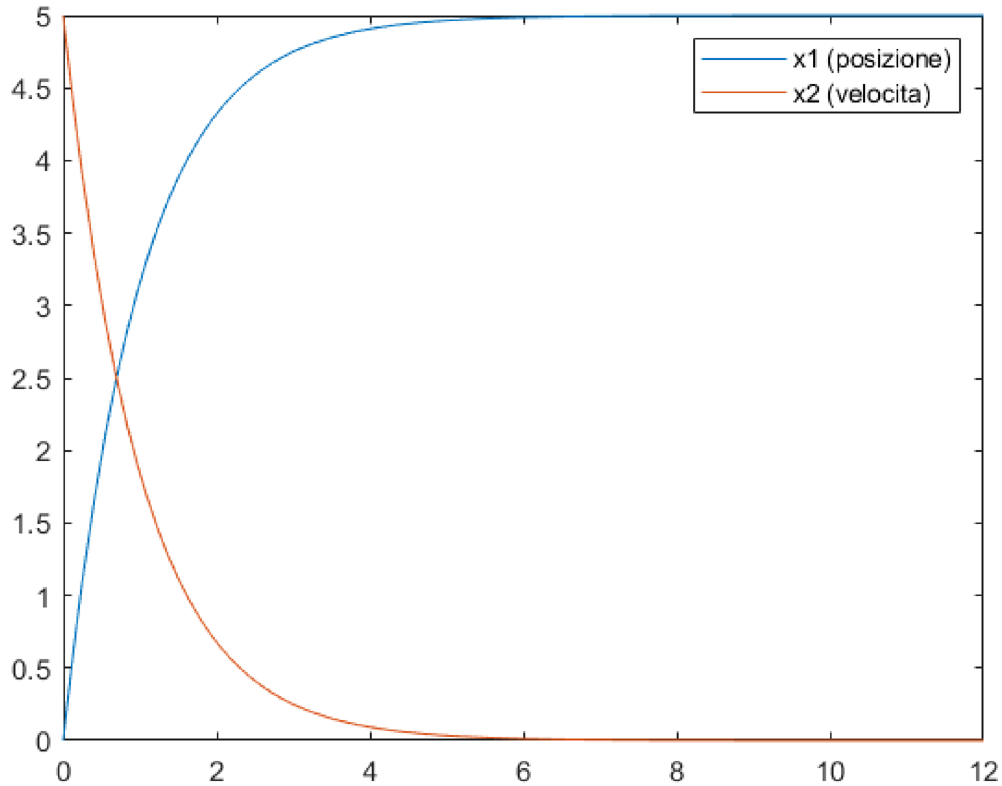
NOTA: il risultato è arrotondato, di default Matlab mostra solo 4 cifre dopo il punto decimale. Per mostrare più cifre:

```
>> format long
>> x_12
x_12 =
   4.99969278938233
   0.00030721061767
```

Soluzione esercizio 2: andamento grafico

- Grafico di funzioni simboliche:

```
>> fplot(eAt*x0,[0 12])
>> legend("x1 (posizione)", "x2 (velocita)")
```



Strumenti numerici e orientati al progetto di sistemi di controllo: Control Systems Tlbx

Control System Toolbox per modelli *state-space*

- 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 (in **formato numerico**!!)

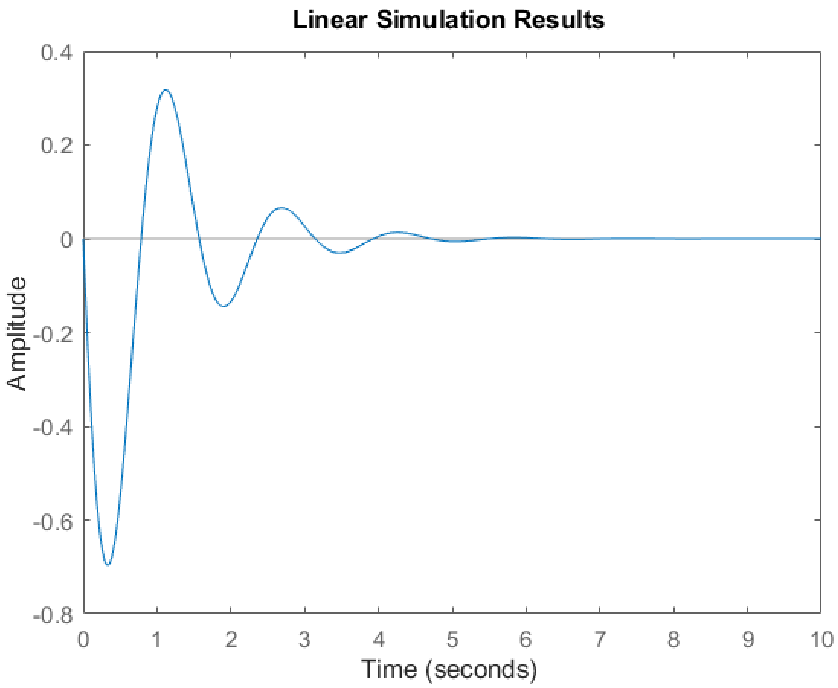
```
>> sys = ss([-1 4;-4 -1],[1;0],[0 1],2)

sys =
  A =
          x1   x2
    x1   -1    4
    x2   -4   -1

  B =
          u1
    x1    1
    x2    0

  C =
          x1   x2
    y1    0    1

  D =
          u1
    y1    2

Continuous-time state-space model.
```

Control System Toolbox per modelli *state-space*

- La funzione **`y=lsim(sys,u,t,x0)`** simula l'evoluzione dello stato del sistema a partire dalle condizioni iniziali, calcolandone <u>uscita (default), tempo simulato e stato</u>

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

NOTA: con l'ultimo comando viene aperta la finestra di analisi grafica per sistemi LTI



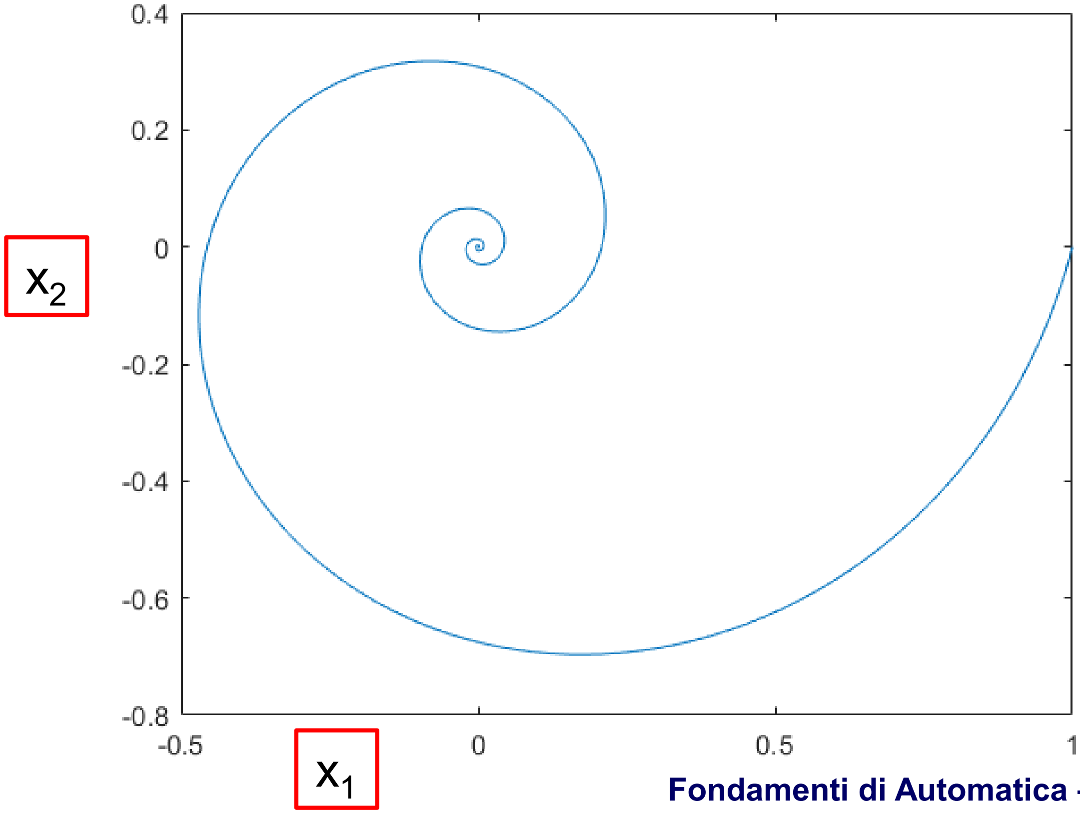
Control Sys tlbx: grafico delle traiettorie dello stato

- Eseguendo il comando con assegnazione dei risultati:

```
>> [y, tout, x]=lsim(sys,u,t,x0);
```

si ottiene anche il vettore di stato completo, se il sistema è di ordine 2 se ne può graficare la ***traiettoria:***

```
>> plot(x(:,1),x(:,2))
```



Control Sys tlbx: grafico delle traiettorie dello stato

- Nota anche la derivata dello stato, si può visualizzare la direzione della traiettoria (meglio con meno "campioni"):

```
>> xn=x(1:5:end,:); un=u(1:5:end);
>> v=(sys.A*xn'+sys.B*un)';
```

$v = \dot{x} = Ax + Bu$

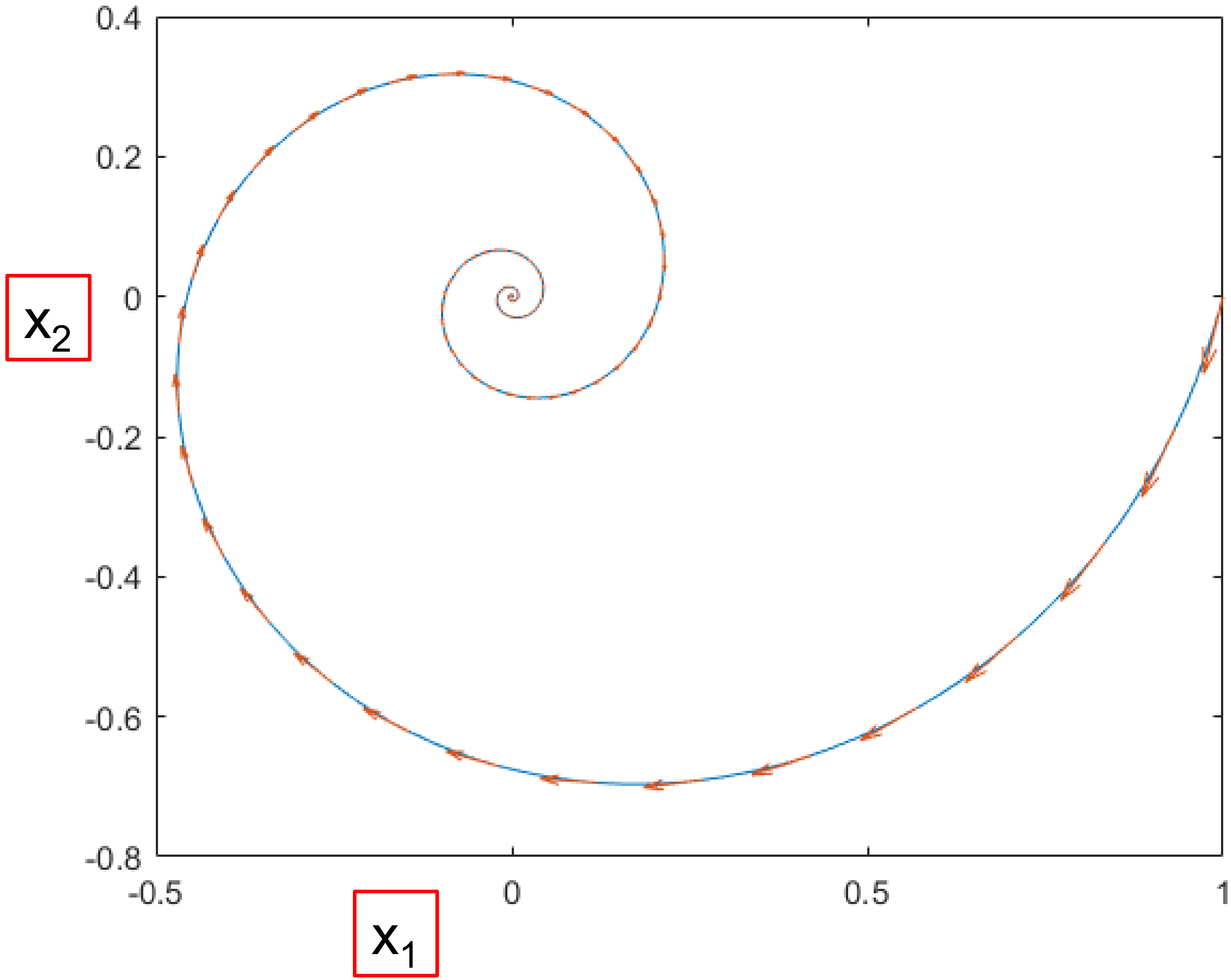
NOTA: le trasposizioni servono solo per eseguire le operazioni in modo corretto su tutti i "campioni" dei vettori di stato e ingresso

```
>> hold on
```

"Fissa" i grafici sulla finestra del plot() precedente!!

```
>> quiver(xn(:,1),xn(:,2),v(:,1),v(:,2))
```

Control Sys tlbx: grafico delle traiettorie dello stato



Strumenti per la soluzione degli esercizi sull’analisi di sistemi:
- controllabilità-raggiungibilità
- osservabilità-ricostruibilità

Matlab: test di controllabilità / osservabilità

- Per verificare se il sistema considerato sia completamente raggiungibile-controllabile è necessario costruire la **matrice di raggiungibilità**:

$$P = \begin{bmatrix} B & AB & A^2B \dots A^{n-1}B \end{bmatrix}$$

  e verificare che abbia **rango = n** (con A nxn)

- Per verificare se il sistema considerato sia completamente osservabile-ricostruibile è necessario costruire la **matrice di osservabilità**:

$$Q = \begin{bmatrix} C & CA & CA^2 \dots CA^{n-1} \end{bmatrix}^T$$

  e verificare che abbia **rango = n** (con A nxn)

Matlab: test di controllabilità / osservabilità

- Grazie al **Control Systems Toolbox**, il test è eseguibile semplicemente lanciando i comandi:

```
>> P=ctrb(A,B)
```

per la matrice di raggiungibilità, poi → `rank(P)`

per il test di controllabilità

```
>> Q=obsv(A,C)
```

per la matrice di osservabilità, poi → `rank(Q)`

per il test di osservabilità

Esercizio riepilogativo (slide 18):

- Proviamo a sostituire Cq=1 e ricavare la matrice A in forma numerica

```
>> Cq=1;
>> A=double(subs(A));
```

- Verificarne la controllabilità (solo **per matrici numeriche**)

```
>> P = ctrb(A,B)
>> rank(P)
```

- Verificarne l'osservabilità (solo **per matrici numeriche, qui effettuata** calcolando la matrice Q trasposta come suggerito nelle dispense..)

```
>> Qt = obsv(A,C)'
>> rank(Qt)
```

INTRODUZIONE A MATLAB

FINE