

# Corso di Laurea Magistrale in Ingegneria Aerospaziale

## A.A. 2012/2013 – II Ciclo

### Esercitazione 3

#### ***Soluzione numerica di Equazioni differenziali Ordinarie (problemi ai valori iniziali) con MATLAB***

#### **1. Integrazione di equazioni differenziali con MATLAB**

L'integrazione di un'equazione differenziale ordinaria o, più in generale di un sistema di equazioni differenziali ordinarie, mediante metodi numerici si realizza in MATLAB attraverso i seguenti passi.

##### **(a) Scrivere l'equazione differenziale**

##### **Esempio1**

$$\begin{cases} y'(t) = t - 2y \\ y(0) = 1 \end{cases} \quad t \in [0,1]$$

L'equazione differenziale viene scritta in una function MATLAB che chiameremo **rhs1.m** con la sintassi:

```
function dydx=rhs1(t,y)
% valuta la parte destra di dy/dx=f(x,y(x))
dydx=t-2*y;
```

I sistemi di equazioni differenziali a valori iniziali si trattano esattamente come le equazioni scalari usando però una sintassi vettoriale nelle funzioni che intervengono.

##### **Esempio 2**

$$\begin{cases} y_1' = y_2 y_3 \\ y_2' = -y_1 y_3 \\ y_3' = -0.51 y_1 y_2 \end{cases} \quad \begin{cases} y_1(0) = 0 \\ y_2(0) = 1 \\ y_3(0) = 1 \end{cases} \quad t \in [0,12]$$

viene scritta in una function MATLAB che chiameremo **rhs2.m** con la sintassi:

```
function dydx=rhs2(t,y)
% valuta la parte destra di dy/dx=f(x,y(x))
dydx=[y(2)*y(3)
      -y(1)*y(3)
      -0.51*y(2)*y(1)];
```

## (b) I metodi numerici

MATLAB fornisce diverse funzioni per la risoluzione di equazioni differenziali ordinarie:

Solver	Problem Type	Order of Accuracy	When to Use
Ode45	Nonstiff	Medium	E' basato su metodi di Runge-Kutta del IV e del V ordine. La scelta di cambiamento del passo è automatica all'interno della funzione, il metodo è piu' accurato di ode23(), impiega di più ad ogni passo ma utilizza passi più ampi. E' la funzione che si raccomanda di usare come primo tentativo per la risoluzione di un nuovo problema.
Ode23	Nonstiff	Low	E' basato su metodi di Runge-Kutta del II e del III ordine. La scelta di cambiamento del passo è automatica all'interno della funzione e dipende dalla tolleranza fissata. Può essere più efficiente di ode45 per tolleranze lasche ed in presenza di una moderata stiffness
Ode113	Nonstiff	Low to high	E' basato su una formula Predictor-Corrector di tipo Adams-Bashforth-Moulton. Può essere più efficiente di ode45 per tolleranze restrittive e quando la funzione f risulta particolarmente costosa. E' un solutore multistep e pertanto richiede la risoluzione numerica per i primi passi. La funzione provvede automaticamente a calcolare i primi passi necessari all'innesco del metodo multistep.
Ode15s	Stiff	Low to medium	E' un solutore di ordine variabile basato su metodi multistep lineari impliciti. E' consigliato quando ode45 risulta troppo lento e si sospetta che il problema sia stiff.
Ode23s	Stiff	Low	E' basato su una formula Rosenbrock di ordine 2. Può essere più efficiente di ode15s per tolleranze lasche.
Ode23t	Moderately Stiff	Low	Utilizza la formula dei trapezi. E' indicato per metodi moderatamente stiff.
Ode23tb	Stiff	Low	Utilizza una combinazione del metodo dei trapezi e metodo BDF di ordine 2

Tabella I: funzioni built-in fornite da MATLAB per la risoluzione di problemi differenziali ai valori iniziali.

Sintassi comune a tutte le funzioni elencate in tabella I:

[t,y]=solutore(@odefun,tspan,y0)

[t,y]=solutore(@odefun,tspan,y0,options,parameters)

dove solutore è uno dei metodi in tabella I.

### Parametri di input:

**odefun** è la funzione che valuta l'equazione in un punto t definita, come descritto in (b)

**tspan** è l'intervallo di integrazione in cui viene calcolata la soluzione. Es: [0,100]

**y0** è il valore iniziale (vettore di valori per un sistema di ODE)

**options** è un vettore che contiene opzioni sulla integrazione numerica, quali tolleranze, passo iniziale, passo massimo,...per maggiori dettagli si veda **help odeset**.

### Parametri di output:

**t** vettore dei punti di discretizzazione dell'intervallo di integrazione

**y** vettore delle soluzioni calcolate.

### Esempio

Per risolvere l'equazione differenziale dell'Esempio1 mediante il metodo ode45():

```
[t,y]=ode45(@rhs1,[0 1],1);
```

Per risolvere l'equazione differenziale dell'Esempio2 mediante il metodo ode45():

```
[t,y]=ode45(@rhs2,[0 12],[0 1 1]);
```

## PASSAGGIO PARAMETRI

Per passare, per esempio i parametri P1,P2, alla funzione ODE definita in rhs.m si modifica la chiamata al solutore come segue:

`[t,y] = ode45(@rhs,tspan,y0,[ ],P1,P2)`

Il solutore chiamerà la funzione rhs(t,y,P1,P2). Eseguire lo script MATLAB **demoODE45args.m** per un'esempio di utilizzo.

## MODIFICA OPZIONI

Se si vogliono cambiare le opzioni di default dei solutori in tabella I occorre usare **odeset** di MATLAB. Un esempio di utilizzo di alcune opzioni è fornito dal file **demoODE45opts.m**. Provare ad eseguirlo modificando le 3 opzioni (tolleranza assoluta e relativa sull'errore locale di troncamento e passi di raffinamento) date come parametri in ingresso.

### (c) Visualizzare la soluzione

Per visualizzare le componenti della soluzione si esegue la funzione

```
%plot della soluzione di una singola equazione differenziale  
plot(t,y)
```

Nel caso di sistemi di equazioni differenziali si eseguirà la funzione plot su ogni colonna della matrice di output y:

```
%plot della prima componente della soluzione (per sistemi)  
plot(t,y(:,1))
```

## 2. Risolvere in MATLAB i seguenti ESERCIZI

1. Realizzare una function **eul\_esp.m** per la risoluzione di equazioni differenziali con il **Metodo di Eulero esplicito**.

In uno script **ex2.m** risolvere l'equazione differenziale

$$\begin{cases} y' = -2t y^2 \\ y(0) = 1 \end{cases}$$

sull'intervallo [0,3], passo h=0.1,utilizzando sia il **Metodo di Eulero esplicito** (**eul\_esp.m**) che il **Metodo di Eulero implicito** e **Heun** forniti (**eul\_imp.m**, **heun.m**).

Ripetere l'esecuzione cambiando il passo h di integrazione in (h= 0.1, 0.25, 0.5, 0.75). Visualizzare le soluzioni approssimate ottenute per valori crescenti del passo h e la soluzione esatta  $y = 1/(1+t^2)$ .

### 2. Applicazione: modello sulla dinamica delle popolazioni.

Si consideri un semplice ecosistema di conigli che hanno un'infinità di cibo e di volpi che si nutrono di conigli per sopravvivere. Un classico modello matematico dovuto a Volterra descrive questo sistema mediante una coppia di equazioni differenziali del primo ordine non lineari:

$$\begin{cases} \frac{dy_1}{dt} = 2y_1 - \alpha y_1 y_2 \\ \frac{dy_2}{dt} = -y_2 + \alpha y_1 y_2 \end{cases} \quad \begin{cases} y_1(0) = r_0 \\ y_2(0) = f_0 \end{cases}$$

dove t è il tempo,  $y_1(t)$  è la popolazione di conigli al tempo t,  $y_2(t)$  la popolazione di volpi al tempo t, e  $\alpha$  è una costante positiva. Quando  $\alpha = 0$  le due popolazioni non interagiscono, e così i conigli

proliferano e le volpi muoiono di inedia. Quando  $\alpha > 0$  le volpi incontrano i conigli con una probabilità che è proporzionale al prodotto dei loro numeri. Tali incontri comportano una riduzione del numero dei conigli e un aumento del numero delle volpi.

- a) Per simulare il sistema, creare una funzione **fox\_rabbit.m** che descrive il sistema di equazioni differenziali (con  $\alpha=0.01$ ).
- b) In uno script **ex3.m** determinare le soluzioni approssimate  $y_1(t)$  ed  $y_2(t)$  ottenute utilizzando i metodi di Eulero esplicito, Eulero implicito e **ode45()** (built in function di MATLAB) per esaminare il comportamento delle popolazioni per  $\alpha=0.01$  e vari valori di  $r_0$  e  $f_0$  definiti in (I), (II), e (III) in un intervallo temporale  $t_0=0$ ,  $t_f=20$ . Si visualizzino quindi tali soluzioni sia con `plot(t, y(:,1), t, y(:,2))` (soluzione rispetto al tempo), sia con `plot(y(:,1), y(:,2))` (soluzione nel piano delle fasi).

(I)  $r_0=2$  e  $f_0=3$ ,

(II)  $r_0=300$  e  $f_0=150$ ,

(III)  $r_0=15$  e  $f_0=22$ .

### OSSERVAZIONI SULL'OUTPUT

Si osservi che il comportamento del sistema è periodico con un periodo prossimo a 5 unità. Con i metodi di Eulero l'orbita `plot(y(:,1), y(:,2))` non si chiude, mentre ciò avviene con `ode45()`. Il metodo di Eulero esplicito richiede un passo piccolo (es  $h=0.01$ ). Qual è il valore  $T_p$  quando entrambe le popolazioni ritornano ai valori originali?

3. Un paracadutista di 80kg si lancia da un aereo ad una altezza di 600m. Dopo 5sec il paracadute si apre. L'altezza in funzione del tempo  $y(t)$  è data da

$$\frac{d^2 y}{dt^2} = -g + a(t)/m \quad y(0) = 600m, \quad \frac{dy}{dt} = 0 \text{ m/s}$$

La resistenza dell'aria  $a(t)$  è proporzionale al quadrato della velocità con differenti costanti di

proporzionalità prima e dopo l'apertura del paracadute.

$$a(t) = \begin{cases} k_1 \left( \frac{dy}{dt} \right)^2, & t < 5s \\ k_2 \left( \frac{dy}{dt} \right)^2, & t \geq 5s \end{cases}$$

Risolvere il problema nello script **ex3.m**. Si consideri il caso  $k_1=1/15$ ,  $k_2=4/15$ . A quale altezza si apre il paracadute? Quanto impiega a raggiungere il suolo? Qual è la velocità di impatto? Visualizzare un grafico di altezza in funzione del tempo.

### 4. Moto di una pallina che rimbalza.

$$\begin{cases} y'_1 = y_2 \\ y'_2 = -9.8 \end{cases} \quad \begin{cases} y_1(0) = 20 \\ y_2(0) = 0 \end{cases}$$

Quando (istante  $t=\tau$ ) la pallina tocca il terreno ( $y(1) = 0$ ) in caduta (direzione negativa) si deve riprendere l'integrazione usando nuovi dati iniziali (attenuazione della velocità):

$$\begin{cases} y_1(\tau) = 0 \\ y_2(\tau) = -0.9 y_2(\tau) \end{cases}$$

- Nello script **motopalla.m** integrare nell'intervallo  $[0,30]$  con la funzione `ode45` e fare il grafico della traiettoria della pallina (ad ogni nuova chiamata della funzione è ragionevole utilizzare l'ultimo passo precedente come passo iniziale con l'opzione 'Initialstep').

Utilizzare l'opzione 'refine' per ottenere un grafico più regolare, con un maggiore numero di punti.

- Ripetere il punto precedente utilizzando la funzione *ode23* per integrare.

### 5. Modello di propagazione della fiamma

Il modello della propagazione della fiamma di un cerino è un esempio di problema stiff, lo trovate nel file **flame.m**. Se accendi un cerino, la fiamma cresce rapidamente fino a che raggiunge una dimensione critica. Quindi rimane ad una certa dimensione poichè la quantità di ossigeno consumata dalla combustione all'interno della fiamma bilancia la quantità disponibile attraverso la superficie. Il modello può essere semplificato nel seguente:

$$\begin{cases} y'(t) = y^2(t) - y^3(t) \\ y(0) = \delta \end{cases}, t \in [0, 2/\delta]$$

la variabile  $y(t)$  rappresenta il raggio della fiamma,  $y^2$  e  $y^3$  sono i parametri della superficie e del volume della bolla. Il parametro critico è il valore iniziale delta che è 'piccolo'. Se delta non è molto piccolo, il problema è moderatamente stiff. Dopo aver calcolato la soluzione esatta con la funzione **dsolve** e averne visualizzato il grafico:

4a) Risolvere il problema con la funzione *ode45*, *reitol*=1.e-4, e i seguenti valori del parametro  $\delta$ :

- 0.01
- 0.0001
- 0.00001
- 0.000001

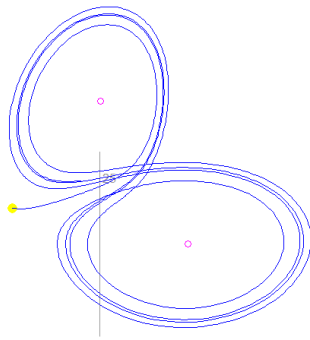
visualizzando, attraverso lo zoom della figura, l'andamento della soluzione in un intorno del valore  $t=1.01/\delta$

4b) risolvere lo stesso problema con un solutore per problemi stiff (stessi valori dei parametri del caso precedente), visualizzando, attraverso lo zoom della figura, l'andamento della soluzione in un intorno del valore  $t=1.05$ .

Riportare in un grafico o in una tabella il numero di passi impiegati dai due solutori (ode 45 e solutore per sistemi stiff) nei due casi..

6. Risolvere il sistema di equazioni di Lorenz,

$$\begin{cases} \frac{dx}{dt} = -ax + ay \\ \frac{dy}{dt} = bx - y - xz, \\ \frac{dz}{dt} = -cz + xy \end{cases} \quad \begin{cases} x(0) = -8 \\ y(0) = 8 \\ z(0) = 27 \end{cases} \quad t \in [0, 20]$$



al variare dei parametri  $a, b, c$ :

$$a=10, b=8/3, c=0.5$$

$$a=10, b=8/3, c=10$$

$$a=10, b=8/3, c=90$$

$$a=10, b=8/3, c=160$$

$$a=10, b=8/3, c=260$$

Per ciascun caso, riportare i grafici dell'attrattore di Lorenz per fase  $(z, x)$  e, in una figura separata, per componenti.