

## 7 Risoluzione numerica di equazioni differenziali ordinarie

**MATLAB** dispone di funzioni specificamente dedicate alla risoluzione numerica di equazioni differenziali ordinarie (**ode**). Da una versione all'altra dell'ambiente di calcolo sono state progressivamente aggiornate le functions originariamente esistenti in questo ambito e sono apparsi nuovi solutori in grado di affrontare problemi di diversa natura (ad esempio i cosiddetti problemi *stiff*). È stata comunque mantenuta un'interfaccia omogenea per tutti i solutori, in modo da consentire all'utente un passaggio immediato da uno all'altro. La collezione delle funzioni **MATLAB** dedicate alla risoluzione delle equazioni differenziali ordinarie è detta **ode suite**. Il comando **odedemo** ne illustra le potenzialità tramite esempi significativi.

Per iniziare, consideriamo il seguente problema di Cauchy:

$$\begin{cases} y^{(n)}(t) = f(t, y, y', \dots, y^{(n-1)}) \\ y(t_0) = y_0 \\ y'(t_0) = y'_0 \\ \vdots \\ y^{(n-1)}(t_0) = y_0^{(n-1)}. \end{cases}$$

Per l'utilizzo di uno qualsiasi dei solutori è necessario riformulare l'equazione nella forma trattata da **MATLAB**, ovvero in *forma normale*, effettuando le seguenti sostituzioni:

$$y_1 = y, y_2 = y', \dots, y_n = y^{(n-1)} \quad (9)$$

in modo da ottenere un sistema di  $n$  equazioni differenziali ordinarie del primo ordine:

$$\begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ \vdots \\ y_n' = f(t, y_1, y_2, \dots, y_n), \end{cases} \quad (10)$$

con condizioni iniziali

$$\begin{cases} y_1(t_0) = y_1^0 \\ y_2(t_0) = y_2^0 \\ \vdots \\ y_n(t_0) = y_n^0. \end{cases}$$

Si consideri ad esempio l'equazione :

$$\begin{cases} y''' - 3y'' - y'y = 0 \\ y(0) = 0, y'(0) = 1, y''(0) = -1. \end{cases} \quad (11)$$

In forma normale essa diventa:

$$\begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ y_3' = 3y_3 + y_2y_1 \end{cases} \quad \text{con} \quad \begin{cases} y_1(0) = 0 \\ y_2(0) = 1 \\ y_3(0) = -1. \end{cases} \quad (12)$$

Per la comprensione della sintassi utilizzata in MATLAB è utile riscrivere il sistema (10) in forma vettoriale:

$$\begin{cases} \mathbf{Y}'(t) = \mathbf{F}(t, \mathbf{Y}) \\ \mathbf{Y}(t_0) = \mathbf{Y}_0, \end{cases} \quad (13)$$

dove si è definito il vettore (colonna):

$$\mathbf{Y} = [y_1, y_2, \dots, y_n]^T = \mathbf{Y}(t).$$

Il sistema scritto nella forma (10), o equivalentemente nella forma compatta (13), deve essere codificato in un file - detto **ode file** - da comporre ad esempio utilizzando l'editor integrato di MATLAB.

Riprendiamo l'esempio (12) e scriviamo il corrispondente **ode file**, che salviamo con il nome myfun.m:

```
function yp=myfun(t,y)
% primo esempio di ode-file
```

```
yp=[y(2)
     y(3)
     3*y(3)+y(2)*y(1)];
```

dove

- nella chiamata alla funzione **myfun** le regole sintattiche prescrivono di passare *entrambi* i parametri **t** e **y** anche quando il sistema, come in questo caso, è autonomo
- il vettore colonna **yp** contiene le componenti del vettore  $\mathbf{F}(t, \mathbf{Y})$
- **y(1)**, **y(2)**, **y(3)** sono le tre componenti della soluzione
- è possibile definire direttamente nella function il vettore delle condizioni iniziali o l'intervallo d'integrazione utilizzati dal solver. È anche possibile passare alla function stessa degli argomenti ulteriori rispetto a quelli strettamente obbligatori **t** e **y**, ad esempio per uno studio parametrico del problema. Si rimanda alla fine di questo paragrafo per l'analisi di questi casi, di meno immediata trattazione.

Ora siamo in grado di utilizzare nel modo più elementare uno qualsiasi dei solver disponibili, mediante la sintassi:

```
[t,y]=nomesolver('myfun',tspan,y0);
```

dove:

- **nomesolver** è il nome del solutore che si vuole utilizzare, ad esempio **ode45**, **ode23**, **ode15s**... (si rimanda alla fine di questo paragrafo e all'**help** per una descrizione più dettagliata di ciascun solver)
- **'myfun'** è la stringa con il nome dell'**ode file**
- **tspan** è il vettore riga contenente gli estremi di integrazione [**t0**, **tf**]
- **y0** è il vettore riga contenente le condizioni iniziali

e dove:

- $\mathbf{t}$  è il vettore colonna contenente i valori discreti  $t_k$  degli “istanti di integrazione”. Si noti che l’intervallo di integrazione  $[\mathbf{t0} \ \mathbf{tf}]$  viene discretizzato dal solver in modo dinamico in funzione dell’errore locale commesso dal metodo, come verrà illustrato più in dettaglio nel seguito
- $\mathbf{y}$  è la matrice di dimensioni  $(\text{size}(\mathbf{t}), \mathbf{n})$  contenente la soluzione  $\mathbf{Y}(t)$  nel formato seguente:

$$\mathbf{y} = \begin{bmatrix} \uparrow & \uparrow & \cdots & \uparrow \\ y_1 & y_2 & \cdots & y_n \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}.$$

Supponiamo di voler integrare il problema (12) nell’intervallo  $[0, 1]$  usando il solver `ode45`:

```
>> [t,y]=ode45('myfun',[0 1],[0 1 -1]);
```

Visualizziamo il risultato (si veda la Figura 31):

```
>> plot(t,y(:,1),'-',t,y(:,2),'--',t,y(:,3),'-.');
```

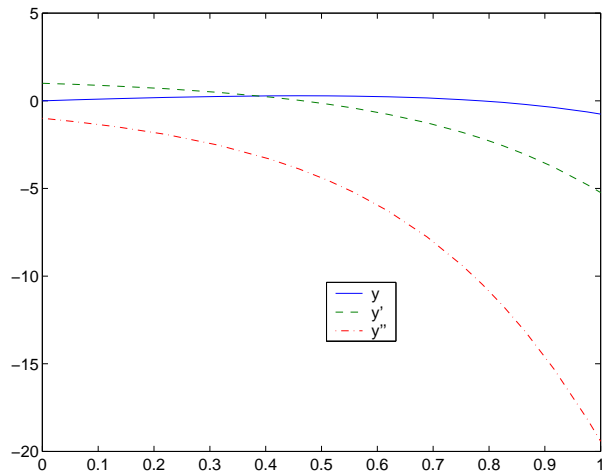


Figura 31: Soluzione del sistema (12) rappresentata componente per componente

**Osservazione.** Per ottenere il valore della soluzione  $\mathbf{Y}$  a *tempi specificati* si usa la sintassi:

```
>> tspan=[t0 t1 t2 ... tf],
```

dove può essere  $\mathbf{t0} > \mathbf{tf}$ , purché i valori intermedi siano ordinati in modo corrispondente. Si osservi che in ogni caso il solver sceglie autonomamente il passo di integrazione temporale e ricostruisce a posteriori i valori della soluzione nei punti  $\mathbf{t1}, \mathbf{t2} \dots$  indicati dall’utente tramite un’opportuna procedura di interpolazione.

**Esempio. Equazione di Van der Pol.** Consideriamo la seguente equazione del secondo ordine:

$$\begin{cases} y'' - \mu(1 - y^2)y' + y = 0 \\ y(0) = 2 \quad y'(0) = 0, \end{cases} \quad (14)$$

dove  $\mu$  è un parametro reale positivo che rappresenta lo smorzamento del sistema. In forma normale essa diventa:

$$\begin{cases} y_1' = y_2 \\ y_2' = \mu(1 - y_1^2)y_2 - y_1 \end{cases} \quad \text{con} \quad \begin{cases} y_1(0) = 2 \\ y_2(0) = 0. \end{cases} \quad (15)$$

Fissiamo  $\mu = 1$  (smorzamento debole) e creiamo l'**ode file** vdpol.m:

```
function yp=vdpol(t,y)
% equazione di Van der Pol
mu=1;
yp=[y(2);mu*(1-y(1)^2)*y(2)-y(1)];
```

Lanciamo il solver ode45 nell'intervallo [0 20] e plottiamo analogamente all'esempio precedente i risultati (si veda la Figura 32 a sinistra):

```
>> [t,y]=ode45('vdpol',[0 20],[2 0]);
>> plot(t,y(:,1),'-',t,y(:,2),'--');
```

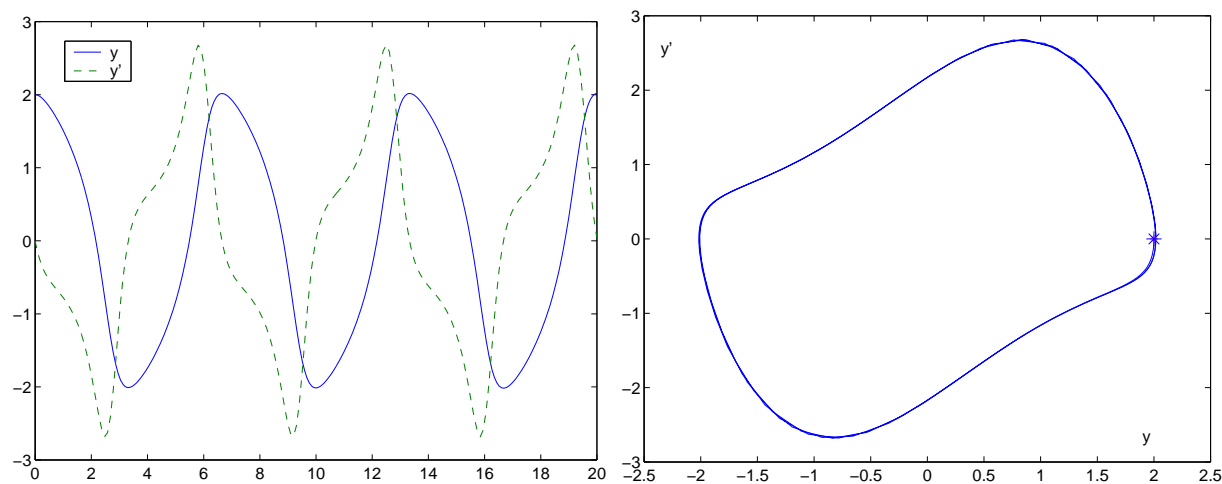


Figura 32: Soluzione dell'equazione di Van der Pol per  $\mu = 1$  rappresentata componente per componente (a sinistra) e nel piano delle fasi (a destra)

È spesso utile rappresentare la soluzione anche nel piano delle fasi (ovvero nel piano  $y, y'$ ), evidenziando con un asterisco la condizione iniziale (si veda la Figura 32 a destra):

```
>> plot(y(:,1),y(:,2))
>> hold
>> plot(2,0,'*')
```

Studiamo ora l'equazione per  $\mu = 1000$  che rappresenta il caso con forte smorzamento: poniamo in vdpol.m  $\mu = 1000$  e rilanciamo il solver, questa volta nell'intervallo di integrazione [0 1000]:

```
>> [t,y]=ode45('vdpol',[0 1000],[2 0]);
```



Osserviamo come il solutore impieghi molto tempo per integrare il sistema di Van der Pol: il problema è diventato *stiff* perché le componenti della soluzione  $\mathbf{Y}$  subiscono variazioni su scale temporali molto diverse tra loro. Il metodo numerico risente di questo squilibrio nelle scale temporali e ne paga il prezzo in termini di una scelta sul passo di integrazione che in alcuni punti risulta molto stringente. In questo caso è opportuno ricorrere alla classe di solver, indicati con il suffisso “s” appositamente studiati per questo genere di problemi, ad esempio i solver `ode15s` e `ode23s`.

```
>> [t,y]=ode15s('vdpol',[0 1000],[2; 0]);
>> plot(t,y(:,1),'o')
```

Ora i tempi di calcolo sono notevolmente diminuiti. Il grafico in scala semilogaritmica delle due componenti della soluzione (mostrato in Figura 33) evidenzia chiaramente le diverse scale temporali caratteristiche di ciascuna componente, da cui la natura *stiff* del problema.

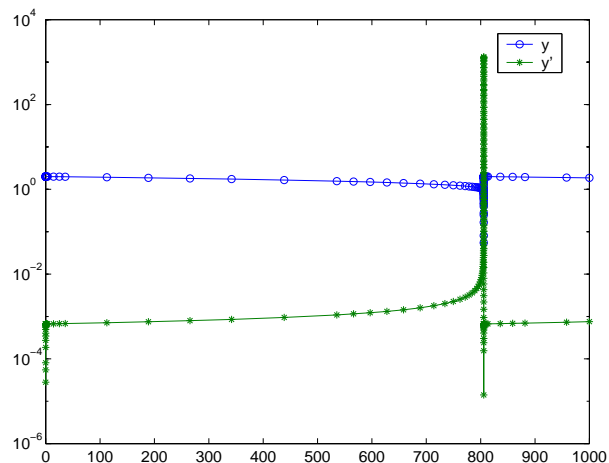


Figura 33: Soluzione dell’equazione di Van der Pol per  $\mu = 1000$  ottenuta con un solver specifico per problemi *stiff*. Si notino le diverse scale di variazione delle due componenti (rappresentate in valore assoluto)

**Esempio. Moto del pendolo senza attrito.** Consideriamo il pendolo di massa  $m$  e lunghezza  $L$  (rappresentato in Figura 34) e studiamo l’evoluzione temporale dell’angolo  $\theta$  formato dal pendolo con la verticale passante per la cerniera a cui esso è vincolato.

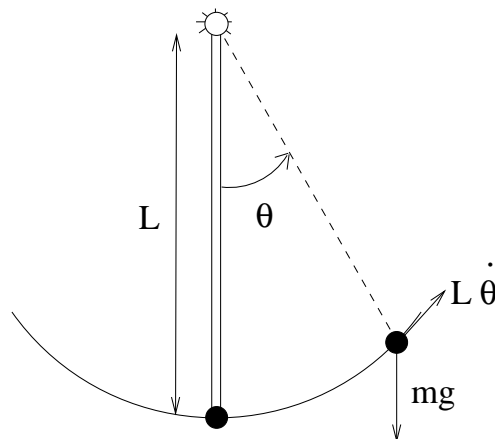


Figura 34: Schematizzazione del problema del pendolo

Si tratta di un sistema ad 1 grado di libertà il cui moto è descritto dalla seguente equazione differenziale ordinaria:

$$\begin{cases} \ddot{\theta} + \omega^2 \sin \theta = 0 & \text{con} \quad \omega^2 = \frac{g}{L} \\ \theta(0) = \theta_0 \\ \dot{\theta}(0) = \dot{\theta}_0, \end{cases} \quad (16)$$

dove si è indicato con il simbolo  $\dot{g}$  la derivata rispetto al tempo della funzione  $g = g(t)$ . Poniamo per semplicità  $\omega^2 = 1$  e scriviamo l'equazione (16) in forma normale:

$$\begin{cases} \dot{y}_1 = y_2 \\ \dot{y}_2 = -\sin(y_1), \end{cases} \quad (17)$$

dove si è posto:

$$\begin{cases} y_1 = \theta \\ y_2 = \dot{\theta} = \dot{y}_1. \end{cases} \quad (18)$$

Il corrispondente **ode file** pendolo.m è:

```
function yp=pendolo(t,y)
%pendolo senza attrito
```

```
yp=[y(2);-sin(y(1))];
```

Risoliamo il problema (17) con `ode45` nell'intervallo  $[0 \ 100]$ , con condizioni iniziali  $\theta_0 = \pi - \delta$ ,  $\dot{\theta}_0 = 0$ . Per  $\delta = 0.01$  (il pendolo si trova quindi inizialmente in alto in posizione quasi verticale con velocità nulla) otteniamo:

```
>> [t,y]=ode45('pendolo',[0 100],[pi-0.01 0]);
>> x=plot(y(:,1),y(:,2))
```

Se confrontiamo il grafico della soluzione calcolata numericamente, riportato in Figura 35 a sinistra, con il grafico della soluzione fisicamente corretta del problema, riportato in Figura 35 a destra, osserviamo che la soluzione numerica *non* è accettabile.

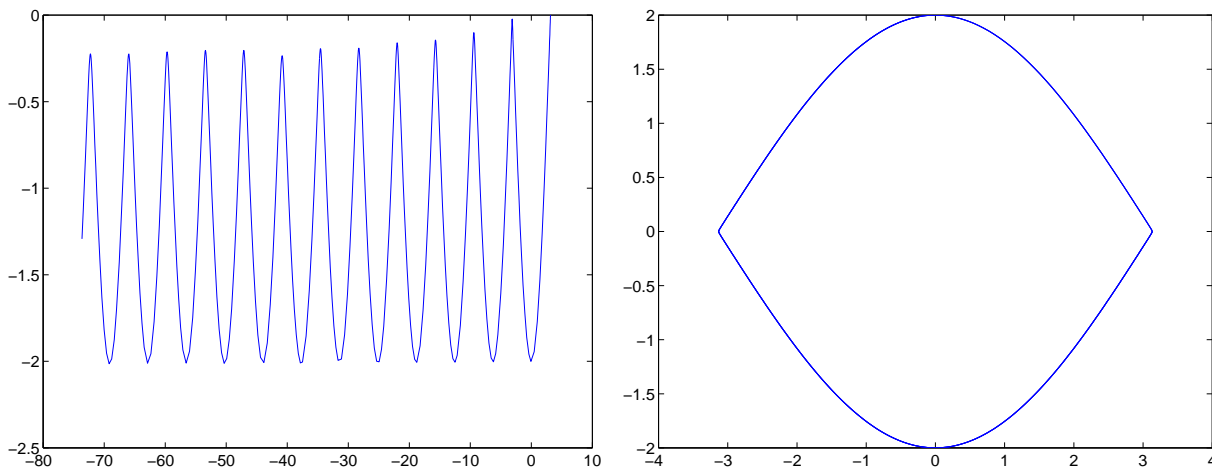


Figura 35: Soluzione nel piano delle fasi per il pendolo con velocità angolare iniziale al di sotto della velocità limite: a sinistra è rappresentata la soluzione fisicamente non accettabile ottenuta con i parametri di default di `ode45` mentre a destra è rappresentata la soluzione corretta ottenuta variando i parametri `RelTol` e `AbsTol`

Infatti per  $\dot{\theta}_0 = 0$  ci troviamo sicuramente al di sotto della velocità limite del pendolo  $\dot{\theta}_{lim} = 2$ , oltre la quale il pendolo compie delle rotazioni complete. Quindi le traiettorie nel piano delle fasi devono essere linee chiuse, ossia il pendolo non riesce a compiere un giro completo ma continua ad oscillare fra l'angolo  $\theta_0 = \pi - \delta$  e  $\theta_f = -(\pi - \delta)$ .

Come possiamo rimediare a questa situazione? Dobbiamo intervenire sulle impostazioni (dette *proprietà*) che determinano la precisione con cui opera il solver.

A questo scopo utilizziamo la struttura `options`: essa viene costruita dal comando `odeset` e viene accettata in ingresso da tutti i solver. La sintassi per assegnare alla proprietà `NomeProp-i` il valore `Val-i` è:

```
options = odeset('NomeProp1',Val1,'NomeProp2',Val2,...)
```

Proviamo a modificare tramite le proprietà `RelTol` e `AbsTol` la tolleranza relativa all'errore locale commesso sulla componente *iesima* della soluzione ad ogni passo di integrazione. MATLAB cerca di soddisfare il seguente criterio sull'errore commesso nell'approssimazione della *i*-esima componente della soluzione  $y(i)$ :

$$|e(i)| \leq \max(\text{RelTol} * \text{abs}(y(i)), \text{AbsTol})$$

```
>> options=odeset('RelTol',1e-6,'AbsTol',1e-7);
>> [t,y]=ode45('pendolo',[0 100],[pi-0.01 0],options);
>> plot(y(:,1),y(:,2));
```

La soluzione ottenuta in questo modo corrisponde al comportamento corretto mostrato a destra in Figura 35.

Esistono numerose altre proprietà su cui è possibile intervenire: per conoscere il significato preciso di ciascuna di esse si rimanda all'`help` di `odeset`. Il solo comando `odeset` senza parametri d'ingresso mostra il valore attuale di ciascuna proprietà (indicato fra parentesi graffe ove vi siano più scelte). Riportiamo il risultato del comando prima di apportare le modifiche indicate sopra (sono quindi indicati fra parentesi graffe i valori di default):

```
AbsTol: [ positive scalar or vector {1e-6} ]
      BDF: [ on | {off} ]
      Events: [ on | {off} ]
InitialStep: [ positive scalar ]
      Jacobian: [ on | {off} ]
      JConstant: [ on | {off} ]
      JPattern: [ on | {off} ]
      Mass: [ {none} | M | M(t) | M(t,y) ]
MassSingular: [ yes | no | {maybe} ]
      MaxOrder: [ 1 | 2 | 3 | 4 | {5} ]
      MaxStep: [ positive scalar ]
NormControl: [ on | {off} ]
      OutputFcn: [ string ]
      OutputSel: [ vector of integers ]
      Refine: [ positive integer ]
      RelTol: [ positive scalar {1e-3} ]
      Stats: [ on | {off} ]
      Vectorized: [ on | {off} ]
```

e dopo di esse:

```

AbsTol: [ positive scalar or vector {1e-7} ]
BDF: [ on | {off} ]
Events: [ on | {off} ]
InitialStep: [ positive scalar ]
Jacobian: [ on | {off} ]
JConstant: [ on | {off} ]
JPattern: [ on | {off} ]
Mass: [ {none} | M | M(t) | M(t,y) ]
MassSingular: [ yes | no | {maybe} ]
MaxOrder: [ 1 | 2 | 3 | 4 | {5} ]
MaxStep: [ positive scalar ]
NormControl: [ on | {off} ]
OutputFcn: [ string ]
OutputSel: [ vector of integers ]
Refine: [ positive integer ]
RelTol: [ positive scalar {1e-6} ]
Stats: [ on | {off} ]
Vectorized: [ on | {off} ]

```

Per uno studio completo del comportamento della soluzione di un'equazione differenziale ordinaria è utile rappresentare le traiettorie percorse dalle componenti della soluzione nel piano delle fasi *al variare delle condizioni iniziali*. Consideriamo ancora il problema del pendolo senza attrito e tracciamo le orbite corrispondenti all'insieme di condizioni iniziali:

$$\begin{cases} \theta_i(0) = 0 \\ \dot{\theta}_i(0) = -3 + i\Delta\dot{\theta}, \end{cases}$$

dove, per  $i = 0, \dots, 12$ , l'incremento  $\Delta\dot{\theta} = 1/2$  fa sì che la velocità angolare iniziale descriva l'intervallo di valori

$$-3 \leq \dot{\theta}_0 \leq 3,$$

in modo da comprendere tutta la gamma di situazioni possibili, ovvero

$$\begin{cases} |\dot{\theta}_0| > \dot{\theta}_{lim} \\ |\dot{\theta}_0| = \dot{\theta}_{lim} = 2\omega = 2 \\ |\dot{\theta}_0| < \dot{\theta}_{lim}. \end{cases}$$

A tale scopo, trovata la soluzione corrispondente ad una certa condizione iniziale, eseguiamo un plot con il comando `hold` per sovrapporre i vari grafici delle traiettorie:

```

clear all
close all
options=odeset('RelTol',1e-6,'AbsTol',1e-7);
hold on
N=12;
deltatp=6/N;
for i=0:N
    tetap=-3+i*deltatp;
    [t,y]=ode45('pendolo',[0 25],[0 tetap],...

```

```

options);
plot(y(:,1),y(:,2),'b',y(:,1),-y(:,2),'b');
end
axis('equal')
grid

```

Per inquadrare meglio la zona di interesse utilizziamo infine il comando `zoom`. Il risultato è mostrato in Figura 36.

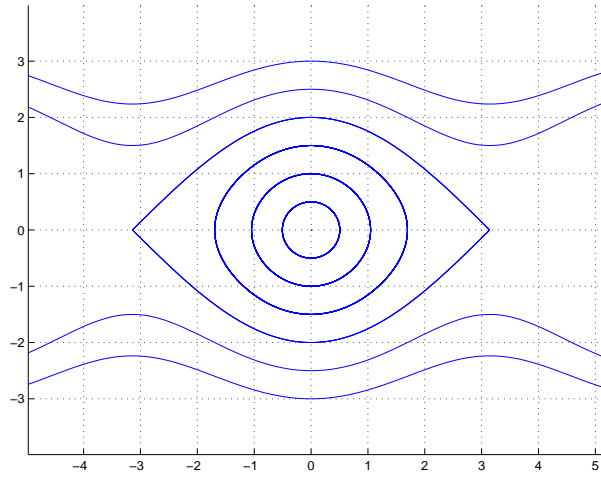


Figura 36: Studio parametrico delle orbite per il problema del pendolo. Le orbite chiuse interne corrispondono a  $|\dot{\theta}_0| < \dot{\theta}_{lim}$ , l'orbita chiusa più esterna corrisponde a  $|\dot{\theta}_0| = \dot{\theta}_{lim} = 2$ , mentre le curve che intersecano l'asse delle ordinate per valori in modulo maggiori di 2 corrispondono a  $|\dot{\theta}_0| > \dot{\theta}_{lim}$ .

**Studio parametrico delle orbite nel piano delle fasi: un esempio più complesso.** Si consideri il seguente problema (tratto da [4], pgg. 312-314): un conduttore AB, di massa  $m = 1$  e lunghezza  $l$  è posto in un piano orizzontale e collegato ad una molla di costante elastica  $k$  che può muoversi perpendicolarmente ad AB, nello stesso piano della molla e di un altro conduttore di lunghezza  $L \gg l$ , posto ad una distanza unitaria dal punto di ancoraggio della molla (si veda la Figura 37).

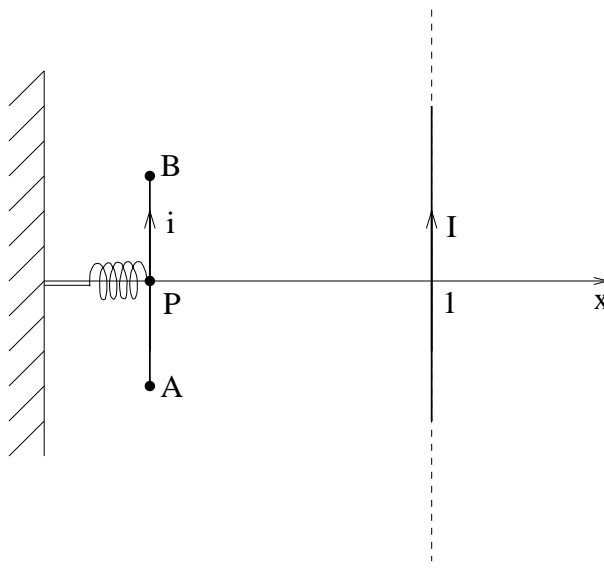


Figura 37: Schematizzazione del problema dello studio del moto del punto P appartenente al conduttore AB

Assumendo che il conduttore AB sia percorso da una corrente  $i$  e che l'altro conduttore sia percorso da una corrente  $I$ , l'equazione di moto del punto P di coordinate  $x = x(t)$  (con  $x = 0$  in assenza di corrente) è:

$$\ddot{x} = -kx + \frac{\lambda}{1-x}, \quad \text{con } \lambda = 2Iil > 0 \quad (19)$$

Studiamo il moto del punto P al variare del parametro  $\lambda$ . Dallo studio qualitativo delle orbite (cf. [4]) si trova che, se  $0 < \lambda < k/4$ , si ottengono i punti critici:

$$\begin{cases} P_\lambda = (p_\lambda, 0) = \left( \frac{1}{2} - \frac{1}{2}\sqrt{1 - 4\frac{\lambda}{k}}, 0 \right) \\ Q_\lambda = (q_\lambda, 0) = \left( \frac{1}{2} + \frac{1}{2}\sqrt{1 - 4\frac{\lambda}{k}}, 0 \right) \end{cases}$$

In questo caso il conduttore AB, partendo da una posizione iniziale tale che  $x_0 < q_\lambda < 1$ , compie infinite orbite periodiche attorno a  $P_\lambda$ , mentre, partendo da una posizione  $q_\lambda < x_0 < 1$ , esso si muove direttamente verso il conduttore grande. Se  $\lambda > k/4$  non vi sono punti critici e il conduttore AB si avvicina direttamente al conduttore grande per ogni scelta della posizione iniziale  $0 < x_0 < 1$ . Infine, per  $\lambda = k/4$  esiste un unico punto critico  $(1/2, 0)$  sul quale vanno a coincidere i punti  $P_\lambda$  e  $Q_\lambda$ . In tale punto degenera sia la velocità che l'accelerazione sono nulle, pertanto il conduttore che vi si trovi inizialmente, permane indefinitamente in quella posizione. Queste considerazioni ci saranno utili nel seguito per interpretare i risultati numerici e giudicare la loro aderenza alla realtà fisica.

Procedendo in maniera analoga agli esempi precedenti riscriviamo l'equazione (19) in forma normale:

$$\begin{cases} \dot{x} = y \\ \dot{y} = -kx + \frac{\lambda}{1-x} \end{cases} \rightarrow \begin{cases} \dot{y}_1 = y_2 \\ \dot{y}_2 = -ky_1 + \frac{\lambda}{1-y_1} \end{cases} \quad (20)$$

e creiamo il file conduttori.m:

```
function yp=conduttori(t,y,flag,k,lambda)
```

```
    yp=[y(2); -k*y(1)+lambda/(1-y(1))];
```

In questo caso, oltre agli usuali parametri obbligatori `t` e `y`, alla function `conduttori` vengono anche passati i parametri `k` e `lambda`. Vedremo tra poco le regole sintattiche alla base del passaggio di questi parametri opzionali. Prima però codifichiamo la chiamata al solver `ode45` nello script `runconduttori.m` che si occupa di gestire la risoluzione del problema per diversi valori della posizione iniziale del punto P, per i tre valori di  $\lambda = (1/8, 1/4, 1/2)$  (avendo fissato  $k = 1$ ), rappresentativi delle tre situazioni possibili illustrate sopra:

```
clear all
close all
options=odeset('RelTol',1e-6,'AbsTol',1e-7);
k=1;
lam=[k/8, k/4, k/2];
x=-0.5:0.1:0.99;
for j=1:length(lam)
    lambda=lam(j);
    figure
    s=sprintf('%7.5f',lambda);
    title(['Lambda = ',s])
    hold on
    for i=1:length(x)
        x0=x(i);
        [t,y]=ode45('conduttori',[0 30],[x0 0],options,k,lambda);
        plot(y(:,1),y(:,2),'b',y(:,1),-y(:,2),'b');
    end
    hold off
end
```

Nelle Figure 38, 39 e 40 sono rappresentate le orbite calcolate numericamente, che corrispondono alle previsioni teoriche. In particolare si osservi nella figura con  $\lambda = k/4$  il punto critico degenere  $(1/2,0)$  evidenziato con un asterisco.

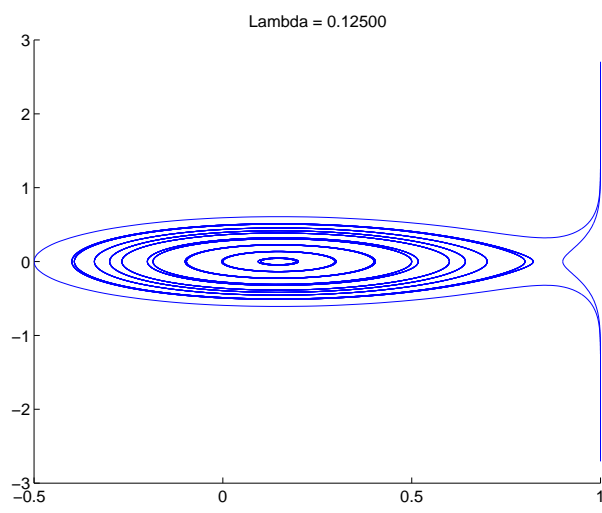


Figura 38: Soluzione del problema del moto del conduttore per  $\lambda = 1/8$

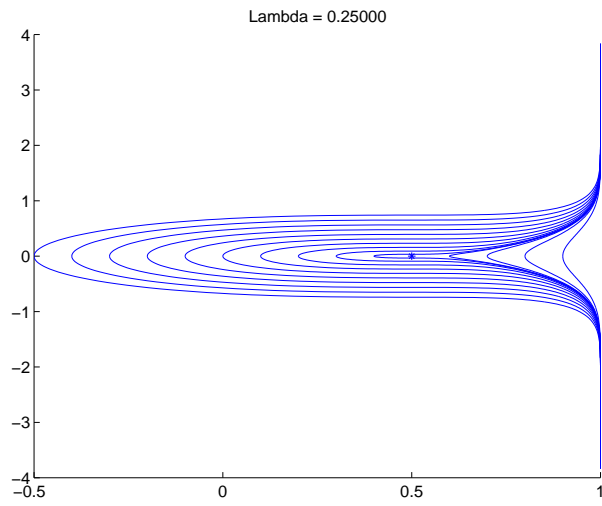


Figura 39: Soluzione del problema del moto del conduttore per  $\lambda = 1/4$

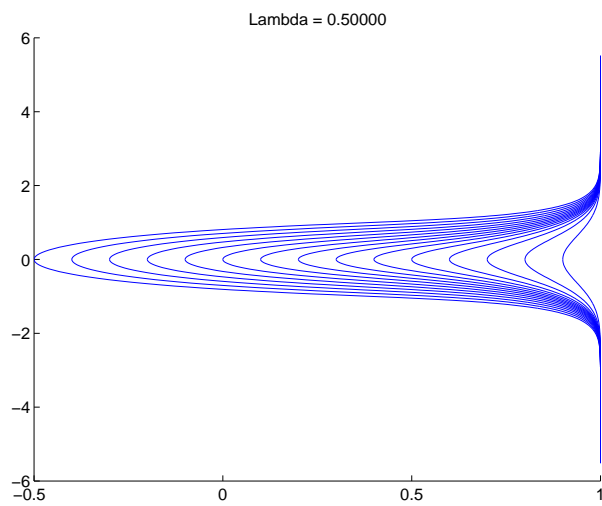


Figura 40: Soluzione del problema del moto del conduttore per  $\lambda = 1/2$



La sintassi *più generale* per chiamare un qualsiasi `ode solver` è:

```
[t,y]=nonesolver('fun',tspan,y0,options,p1,p2,p3,...),
```

dove `p1,p2,p3` sono dei parametri opzionali da passare alla function (nel nostro caso passiamo `k` e `lambda`). Osserviamo che obbligatoriamente i parametri `p1,p2,p3...` devono essere passati *per ultimi* nella chiamata al solver. Qualora una o più delle variabili `tspan`, `y0`, `options` sia definita nell'**ode file** e quindi non sia assegnata dall'esterno (oppure nel caso in cui `options` non sia definita perché si accettano i parametri standard) è comunque necessario rispettare l'ordine di passaggio dei parametri indicando con il simbolo `[ ]` la variabile mancante. Supponiamo ad esempio di aver definito il vettore delle condizioni iniziali `y0` nell'**ode file**: in questo caso la sintassi generale vista sopra diventa:

```
[t,y]=nonesolver('fun',tspan,[ ],options,p1,p2,p3,...),
```

dove abbiamo contrassegnato il posto spettante ad `y0` con il simbolo `[ ]`. Se tutti e tre i parametri `tspan`, `y0`, `options` fossero definiti nell'**ode file**, andrebbe utilizzata la sintassi:

```
[t,y]=nonesolver('fun',[ ],[ ],[ ],p1,p2,p3,...).
```

Ad esempio, avendo definito la function:

```
function yp=pendolo(t,y)
%pendolo senza attrito
```

```
tspan=[0 100];
yp=[y(2);-sin(y(1))];
```

utilizziamo la chiamata:

```
>> [t,y]=ode45('pendolo',[ ],[pi-0.01 0],options);
```

Per concludere questo paragrafo dedicato alla risoluzione numerica in **MATLAB** delle equazioni differenziali alle derivate ordinarie, presentiamo una breve panoramica dei solver disponibili e delle caratteristiche di ciascuno di essi. È opportuno osservare come solo la conoscenza dello specifico problema da analizzare, combinata con l'esperienza, permettano di determinare in ciascun caso la scelta più favorevole.

| <i>solver</i>  | <i>metodo implementato</i>              | <i>osservazioni</i>                                   |
|----------------|---|---|
| <b>ode45</b>   | Runge-Kutta 4-5 esplicito ad un passo   | da usare come primo tentativo                         |
| <b>ode23</b>   | Runge-Kutta 2-3 esplicito ad un passo   | tipicamente più efficiente di <b>ode45</b>            |
| <b>ode113</b>  | Adams-Bashforth-Moulton                 | consigliabile se la valutazione di <b>F</b> è costosa |
| <b>ode15s</b>  | multistep implicito di ordine variabile | da usare come primo tentativo                         |
| <b>ode23s</b>  | solutore ad un passo                    | tipicamente più efficiente di <b>ode15s</b>           |
| <b>ode15s</b>  | multistep implicito di ordine variabile | da usare come primo tentativo                         |
| <b>ode23s</b>  | solutore ad un passo                    | tipicamente più efficiente di <b>ode15s</b>           |
| <b>ode23t</b>  | schema basato sul metodo del trapezio   | non dissipativo                                       |
| <b>ode23tb</b> | trapezio + differenze all'indietro      | equivalente a Runge-Kutta implicito                   |

Tabella 3: Descrizione e caratteristiche degli **ode solver**. I solver con suffisso “s” sono specifici per problemi *stiff*

## 8 Trasformata Rapida di Fourier (FFT)

In questo paragrafo verrà illustrato in maniera elementare l'uso della function `fft` per la trasformata rapida di Fourier (in inglese, Fast Fourier Transform, FFT), che ha grande importanza in numerosi applicazioni scientifiche ed ingegneristiche. Per una descrizione dettagliata dell'algoritmo e della sua implementazione numerica si rimanda ad esempio a [2].

Supponiamo di aver schematizzato il moto di un sistema meccanico come il moto di un punto di massa  $m$  sottoposto all'azione di richiamo elastico di una molla di rigidezza  $k$  incognita (si veda la Figura 41).

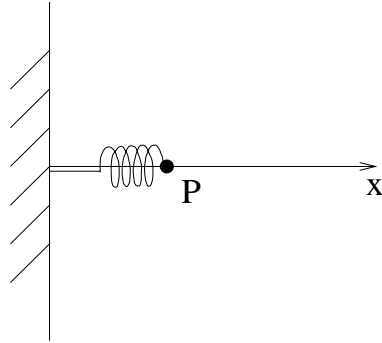


Figura 41: Schematizzazione del sistema massa-molla

L'equazione di moto del punto è:

$$\ddot{x} + \omega^2 x = 0 \quad \text{dove} \quad \omega^2 = \frac{k}{m}$$

La posizione e la velocità iniziali del punto siano rispettivamente  $x(0) = 2\pi/16$  e  $\dot{x}(0) = 0$ . La soluzione esatta di moto armonico è:

$$x = x(0)\cos(\omega t)$$

Si vuole calcolare il valore della costante di rigidezza della molla a partire dalla misurazione della posizione  $x$  del punto, affetta da rumore. Supponiamo che le misure siano effettuate nell'intervallo:

```
>> tspan=[0:2^(-7):128-2^(-7)];
```

ovvero con una frequenza di circa 130Hz (ovvero di  $1/(2^{-7})$ Hz). Costruiamo artificialmente il segnale `xvn` sovrapponendo ad un segnale con un contenuto in frequenza di 50Hz, che chiamiamo `xv`, del rumore casuale a media nulla:

```
>> omega=2*pi*50;
>> x0=2*pi/16;
>> N=length(tspan);
>> f=128/N*(0:N/2-1);

>> xv=x0*cos(omega*tspan);
>> xvn=xv+4*randn(size(tspan));
```

Osserviamo come entrambi i vettori **tspan** e **f** siano stati suddivisi in un numero di intervalli pari ad una opportuna potenza di 2 ( $2^7$  intervalli nel nostro esempio). Questa scelta (che non è in generale obbligatoria) consente di ottimizzare l'efficienza delle prestazioni dell'algoritmo numerico della FFT.

Analizziamo il segnale misurato nel dominio delle frequenze calcolando la trasformata di Fourier del vettore **xvn**. In generale, dato un vettore  $x$  avente  $N$  componenti, il comando **fft(x)** calcola i coefficienti dello sviluppo di Fourier secondo la relazione:

$$\hat{X}(k) = \sum_{j=1}^N x(j) \omega_N^{(j-1)(k-1)},$$

dove abbiamo definito:

$$\omega_N = e^{-2\pi/N}.$$

```
>> XVN=fft(xvn);
```

Calcoliamo la densità di energia spettrale che è una misura dell'energia contenuta in ogni frequenza come:

$$P = \sum_{k=1}^N \frac{(\hat{X}(k) * \overline{\hat{X}(k)})}{N}$$

```
>> PVN=XVN.*conj(XVN)/N;
>> plot(f,PVN(1:N/2))
```

Valutiamo con il comando **max** il valore della frequenza dove si registra il massimo della densità di energia spettrale (chiaramente sarebbe necessario utilizzare delle tecniche di filtraggio, in questo contesto ci limitiamo a mostrare qualitativamente il risultato):

```
>> [y,j]=max(PVN);
```

```
>> f(j)
```

```
ans =
```

```
50
```

Il massimo di energia viene registrato per una frequenza pari a 50Hz (si veda la Figura 42). Ricordando che  $f = \omega/2\pi$ , possiamo ricavare il valore di  $\omega$  e da esso il valore della rigidità equivalente  $k$ .

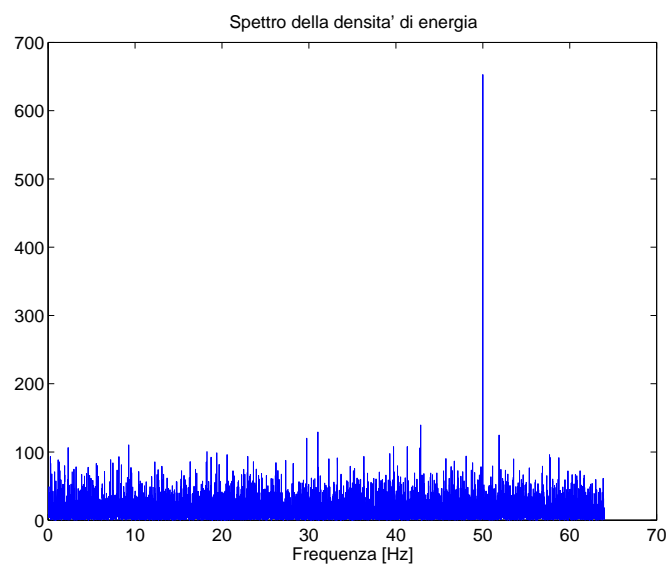


Figura 42: Spettro della densità di energia del segnale.

## 9 Il toolbox pde

In questo paragrafo illustriamo il toolbox di MATLAB pde. Il toolbox fornisce uno strumento molto potente e flessibile per la soluzione di problemi descritti da equazioni alle derivate parziali, in due dimensioni spaziali e in tempo, quindi con condizioni al contorno e iniziali. Per la discretizzazione del problema si adotta il metodo numerico degli Elementi Finiti (FEM). Il toolbox può essere utilizzato sia in modalità grafica, tramite una Interfaccia Grafica (GUI) ad un ambiente autocontenuto oppure accedendo alle singole funzioni dalla linea di comando. La GUI è attivabile con il comando `pdetool`. In particolare sono supportati i seguenti tipi di problemi

- *ellittico*:

$$-\operatorname{div}(c\nabla u) + au = f$$

- *parabolico*:

$$d\frac{\partial u}{\partial t} - \operatorname{div}(c\nabla u) + au = f$$

- *iperbolico*:

$$d\frac{\partial^2 u}{\partial t^2} - \operatorname{div}(c\nabla u) + au = f$$

- *autovalori*:

$$-\operatorname{div}(c\nabla u) + au = \lambda du$$

dove i coefficienti  $c$ ,  $a$ ,  $f$ ,  $d$  e la soluzione  $u$  sono funzioni scalari a valori complessi,  $c$  può essere una matrice  $2 \times 2$  di funzioni e  $\lambda$  è l'autovalore incognito. Nel caso parabolico e iperbolico i coefficienti  $c$ ,  $a$ ,  $f$  e  $d$  possono dipendere anche dal tempo. È anche disponibile un solutore non lineare per il problema ellittico

$$-\operatorname{div}(c(u)\nabla u) + a(u)u = f(u)$$

ove  $c$ ,  $a$  e  $f$  sono funzioni della soluzione  $u$ . Tutti i solutori sono in grado di trattare il caso vettoriale di dimensione 2, cioè in cui la soluzione è il vettore di funzioni  $(u_1, u_2)$  che soddisfano

$$\begin{cases} -\operatorname{div}(c_{11}\nabla u_1) - \operatorname{div}(c_{12}\nabla u_2) + a_{11}u_1 + a_{12}u_2 = f_1 \\ -\operatorname{div}(c_{21}\nabla u_1) - \operatorname{div}(c_{22}\nabla u_2) + a_{21}u_1 + a_{22}u_2 = f_2. \end{cases}$$

Dalla linea di comando possono essere risolti problemi vettoriali di dimensione arbitraria  $> 2$ . Nel caso ellittico, è implementato un algoritmo adattativo di raffinamento della mesh di calcolo che può essere usato insieme con il solutore non lineare. Inoltre è anche disponibile un risolutore veloce per l'equazione di Poisson (Fast Poisson Solver) su griglie rettangolari.

### Osservazione

La classificazione precedente è puramente *formale* e non è necessario che i dati e le condizioni al contorno rendano il problema ellittico, parabolico o iperbolico da un punto di vista strettamente matematico.

Indichiamo con  $\Omega$  il dominio in cui viene risolto il problema e con  $\partial\Omega$  il suo bordo. Sono definite le seguenti condizioni al contorno per lo scalare  $u$ :

- *Dirichlet*:  $hu = r$  sul bordo  $\partial\Omega$ .
- *Neumann Generalizzata (Robin)*:  $c\frac{\partial u}{\partial n} + qu = g$  su  $\partial\Omega$ , ove  $n$  rappresenta la direzione normale uscente da  $\partial\Omega$ ,  $g$ ,  $q$ ,  $h$  e  $r$  sono funzioni a valori complessi definite su  $\partial\Omega$ . Il problema agli autovalori è omogeneo, cioè si ha  $g = 0$  e  $r = 0$ . Nel caso non lineare i coefficienti  $g$ ,  $q$ ,  $h$  e  $r$  possono dipendere da  $u$ , mentre nel caso parabolico e iperbolico essi possono anche dipendere dal tempo. Nel caso vettoriale la condizione di Dirichlet è

$$\begin{cases} h_{11}u_1 + h_{12}u_2 = r_1 \\ h_{21}u_1 + h_{22}u_2 = r_2 \end{cases}$$

la condizione di Neumann generalizzata è

$$\begin{cases} c_{11}\frac{\partial u_1}{\partial n} + c_{12}\frac{\partial u_2}{\partial n} + q_{11}u_1 + q_{12}u_2 = g_1 \\ c_{21}\frac{\partial u_1}{\partial n} + c_{22}\frac{\partial u_2}{\partial n} + q_{21}u_1 + q_{22}u_2 = g_2 \end{cases}$$

mentre la condizione mista è

$$\begin{cases} h_{11}u_1 + h_{12}u_2 = r_1 \\ c_{11}\frac{\partial u_1}{\partial n} + c_{12}\frac{\partial u_2}{\partial n} + q_{11}u_1 + q_{12}u_2 = g_1 + h_{11}\mu \\ c_{21}\frac{\partial u_1}{\partial n} + c_{22}\frac{\partial u_2}{\partial n} + q_{21}u_1 + q_{22}u_2 = g_2 + h_{12}\mu \end{cases}$$

dove  $\mu$  è calcolato in modo tale da soddisfare la condizione di Dirichlet.

Il toolbox può essere utilizzato per modellare un'ampia gamma di fenomeni in tutti i rami dell'ingegneria e delle scienze. Le equazioni ellittiche e paraboliche sono utilizzate per descrivere, ad esempio

- fenomeni di trasporto del calore nei materiali in condizioni stazionarie e non stazionarie
- problemi di diffusione e di trasporto in mezzi porosi
- elettrostatica in mezzi conduttori e dielettrici
- flussi potenziali

L'equazione iperbolica è usata per

- propagazione di onde in regime transitorio o armoniche in acustica e in elettromagnetismo
- spostamento trasversale di membrane

I problemi agli autovalori hanno applicazione, ad esempio, per la

- determinazione gli stati di vibrazione naturali in membrane e in problemi di meccanica strutturale.

Per andare incontro a queste diverse esigenze il toolbox dispone di otto interfacce ai campi di applicazione seguenti

- Meccanica strutturale (sforzi piani)
- Meccanica strutturale (deformazioni piane)
- Elettrostatica
- Magnetostatica
- Elettromagnetismo in regime di corrente alternata
- Mezzi conduttori in regime di corrente continua
- Conduzione del calore
- Diffusione

Queste applicazioni possono essere selezionate tramite l'opzione *Options* e poi *Application* del menu della GUI. Infine, il toolbox può essere utilizzato per scopi didattici come complemento per la comprensione della teoria del metodo degli Elementi Finiti.

La descrizione sarà molto breve e si limiterà alla soluzione passo-passo del problema ellittico con condizioni al contorno miste Dirichlet/Neumann

$$\begin{cases} -\Delta u = f & \text{in } \Omega \\ u = g_0 & \text{su } \Gamma_0 \\ \frac{\partial u}{\partial n} = g_1 & \text{su } \Gamma_1 \end{cases}$$

dove  $\frac{\partial u}{\partial n}$  è la derivata nella direzione normale uscente da  $\partial\Omega$ . Assumeremo  $\Omega = (-1, 1)^2$ ,  $\Gamma_1 = \{x = -1, -1 \leq y \leq 1\}$ ,  $\Gamma_0 = \partial\Omega \setminus \Gamma_1$ ,  $f = 1$ ,  $g_0 = 0$  e  $g_1 = \sin(\pi x)$ . Il problema considerato non ha la pretesa di avere una significativa interpretazione fisica ma è sufficientemente rappresentativo di un uso complesso del toolbox.

Incominciamo la sessione di lavoro digitando il comando `pdetool`. MATLAB aprirà la finestra della GUI che mostriamo in Figura 43.

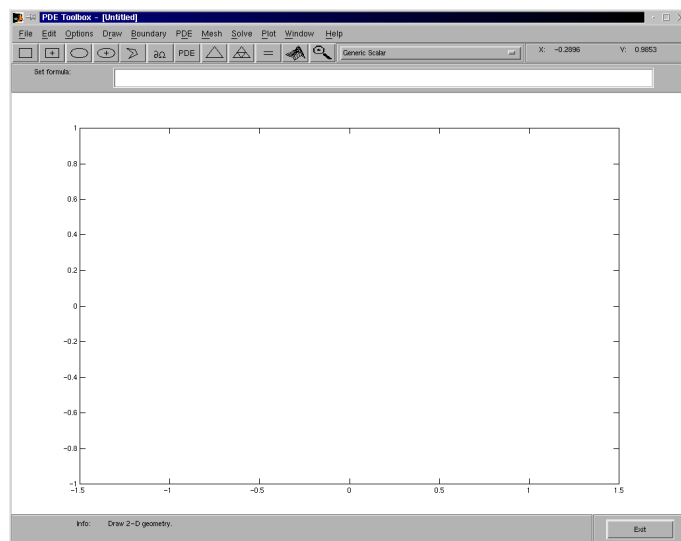


Figura 43: Finestra d'avvio di pdetool

La sessione di lavoro si divide tipicamente in sei fasi:



1. descrizione del dominio (Draw Mode);
2. descrizione delle condizioni al contorno (Boundary Mode);
3. descrizione del problema (PDE Mode);
4. generazione della mesh (Mesh Mode);
5. soluzione del problema (Solve PDE);
6. visualizzazione della soluzione (Plot Solution).

Queste procedure sono accessibili dal menu della finestra sotto le opzioni *Draw*, *Boundary*, *PDE*, *Mesh*, *Solve* e *Plot* rispettivamente, oppure dalla toolbar appena sottostante e riportata in Figura 44.



Figura 44: Toolbar

Nei prossimi paragrafi descriveremo ciascuna di queste fasi.

## 9.1 Draw Mode

In questa fase si definisce il dominio del problema. Clicchiamo sull'opzione *Draw* e dal menu a tendina selezioniamo *Rectangle/square*.

Oppure selezioniamo direttamente l'icona a destra. Portiamo il cursore sul punto di coordinate  $(-1,1)$  e tenendo premuto il tasto sinistro del mouse, portiamo il cursore sul punto di coordinate  $(1,1)$  e poi rilasciamo. Si ottiene la situazione rappresentata in Figura 45.

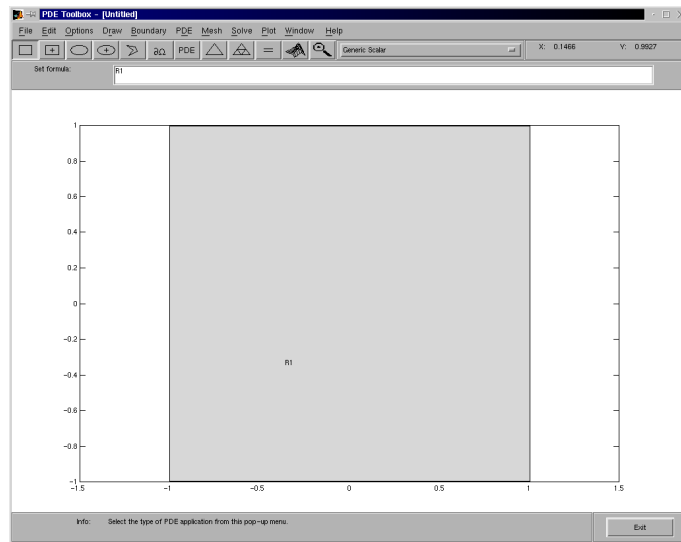


Figura 45: Definizione del dominio

MATLAB crea un *oggetto rettangolo* identificato da  $R_1$  che ha determinate proprietà, precisamente: ascissa (*Left*) e ordinata (*Bottom*) del punto inferiore sinistro, larghezza (*Width*), altezza (*Height*) e nome (*Name*). La GUI offre quattro tipi di oggetti solidi:

poligoni, rettangoli, cerchi e ellissi. Gli oggetti sono usati per creare un modello di *Geometria Solida Costruttiva* (CSG). Ad ogni oggetto viene assegnata una etichetta e con l'uso dell'algebra degli insiemi la geometria risultante può essere ottenuta per combinazioni di unioni, intersezioni e differenze di insiemi. Per default, la CSG costruisce l'unione di tutti gli oggetti definiti dall'utente. Al nostro rettangolo viene assegnata l'etichetta  $R_1$ . Possiamo modificare le proprietà del rettangolo facendo doppio-click su un suo punto. Appare una finestra di dialogo dalla quale è possibile ridefinire le coordinate del rettangolo. In particolare la situazione finale corretta è rappresentata in Figura 46. Si termina cliccando su *OK*.

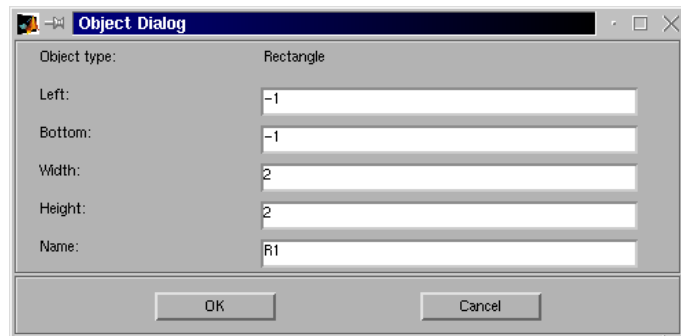


Figura 46: Definizione precisa della geometria del rettangolo

È possibile salvare il modello CSG come M-file. Per lo scopo si usa l'opzione *Save As...* dal menu *File* e si inserisce un nome. È bene salvare a intervalli regolari il proprio modello usando l'opzione *Save*. Tutti i salvataggi successivi verranno effettuati sullo stesso file.

## 9.2 Boundary Mode

Siamo pronti per definire le condizioni al contorno per il problema considerato. A questo scopo selezioniamo l'opzione *Boundary* dal menu e clicchiamo su *Boundary Mode* oppure selezioniamo direttamente l'icona a destra.



I bordi del rettangolo diventano delle frecce orientate di colore rosso. Il colore riflette il tipo di condizione al contorno e la freccia è diretta verso l'estremo finale del segmento. L'informazione sulla direzione è fornita per dare la possibilità di parametrizzare la condizione al contorno lungo il bordo (ad esempio, nel caso di bordi curvilinei). La condizione al contorno può essere (tipicamente) una funzione di  $x$  e  $y$  o semplicemente una costante. La condizione di default è di tipo Dirichlet:  $u = 0$  sul bordo. Il colore rosso indica una condizione di Dirichlet, il blue identifica una condizione di Neumann mentre il verde indica una condizione di tipo misto. Clicchiamo sul lato sinistro due volte per selezionarlo. Si apre una finestra di dialogo e da essa impostiamo *Neumann*. La condizione che possiamo specificare, come abbiamo visto, è del tipo  $c \frac{\partial u}{\partial n} + qu = g$ . Nel nostro caso si ha  $q = 0$  e  $g = \sin(\pi x)$  ( $c$  verrà definito nella fase successiva). Per impostare quest'ultimo valore occorre utilizzare la sintassi *MATLAB* usuale, con le operazioni "puntate", eventualmente. Nel nostro caso si scriverà `sin(pi*x)` nell'apposita riga. Sui rimanenti tre lati dobbiamo imporre la condizione di Dirichlet. A tale fine clicchiamo in sequenza sui tre lati tenendo premuto il tasto *Shift* della tastiera: i tre lati assumeranno un colore nero. Clicchiamo due volte sull'ultimo lato selezionato, sempre premendo il tasto *Shift* per aprire la finestra di dialogo. Nel nostro caso accettiamo il default  $h = 1$  e  $r = 0$  premendo *OK*.

## 9.3 PDE Mode

Ora dobbiamo impostare il tipo di problema. A questo scopo selezioniamo l'opzione *PDE* dal menu e poi la voce *PDE Specification* oppure selezioniamo direttamente l'icona a destra.



Si apre una finestra di dialogo che ci offre la possibilità di specificare il tipo di problema (ellittico, parabolico, iperbolico o agli autovalori) con i relativi coefficienti. Nel nostro caso è sufficiente accettare il default dopo avere modificato il valore di  $f$  da 10 a 1. La situazione finale è rappresentata in Figura 47.

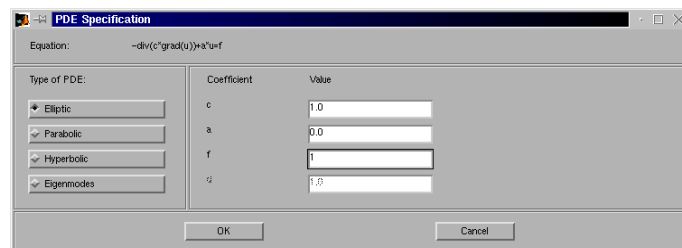


Figura 47: Scelta del problema e definizione dei dati

## 9.4 Mesh Mode

Procediamo alla discretizzazione del dominio generando la cosiddetta triangolazione. Si ricorda infatti che il metodo numerico utilizzato per l'approssimazione della PDE è il metodo degli elementi finiti. Per maggiori informazioni su tale metodo rimandiamo a [3], Cap. 12. Basti sapere che la soluzione  $u$  del problema viene approssimata con una funzione  $u_h$  continua su  $\Omega$  e lineare su ogni triangolo i cui gradi di libertà sono i valori assunti in corrispondenza dei nodi della triangolazione. Per generare la triangolazione (mesh) selezioniamo l'opzione *Mesh* dal menu e poi *Initialize Mesh* oppure clicchiamo direttamente sull'icona a destra.



Verrà generata la mesh rappresentata in Figura 48.

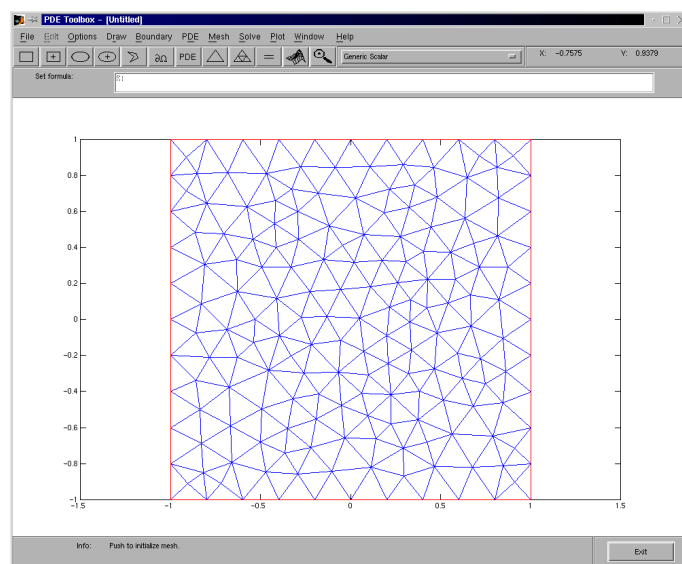


Figura 48: Mesh iniziale

Allo scopo di ottenere una soluzione di qualità migliore, possiamo decidere di raffinare

la mesh, cioè di generare triangoli più piccoli. Clicchiamo quindi sull'icona a destra e otterremo il risultato mostrato in Figura 49.

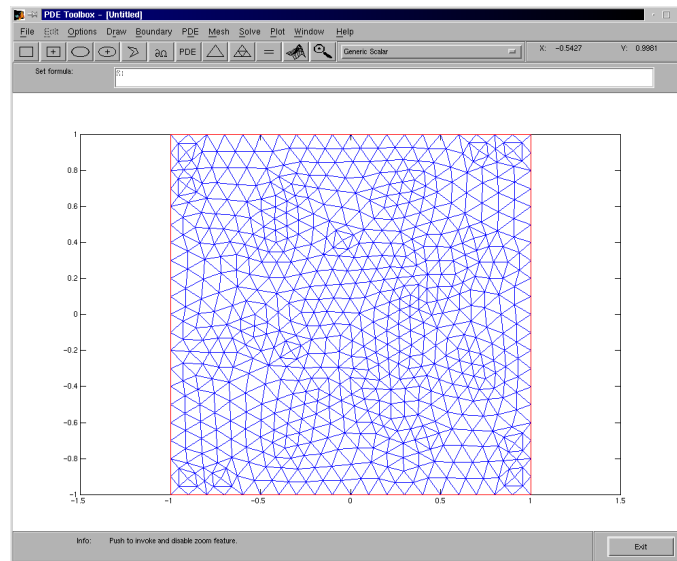


Figura 49: Mesh raffinata

Possiamo inoltre migliorare la qualità della mesh tramite l'opzione *Jiggle Mesh* del menu e modificare altri parametri che controllano la generazione della mesh selezionando *Parameters* dal menu *Mesh*. Per annullare ogni modifica si seleziona l'opzione *Undo Mesh Change*, sempre dal menu *Mesh*.

## 9.5 Solve PDE

Il metodo degli elementi finiti trasforma il problema differenziale di partenza che ha dimensione infinita in un problema a dimensione finita dipendente dal numero di nodi della mesh di calcolo. Precisamente, se il problema differenziale di partenza è lineare si ottiene un sistema lineare algebrico mentre se il problema differenziale è non lineare si perviene a un sistema algebrico non lineare. In entrambi i casi, le incognite sono i valori della funzione  $u_h$  nei nodi della mesh. Per risolvere tale sistema lineare (non lineare) selezioniamo *Solve* da menu e poi *Solve PDE* oppure clicchiamo direttamente l'icona a destra.



MATLAB visualizza automaticamente la soluzione tramite un grafico bidimensionale a colori in cui la gradazione del colore varia linearmente con i valori assunti dalla soluzione numerica che è interpolata linearmente sui triangoli. Il risultato è rappresentato in Figura 50.

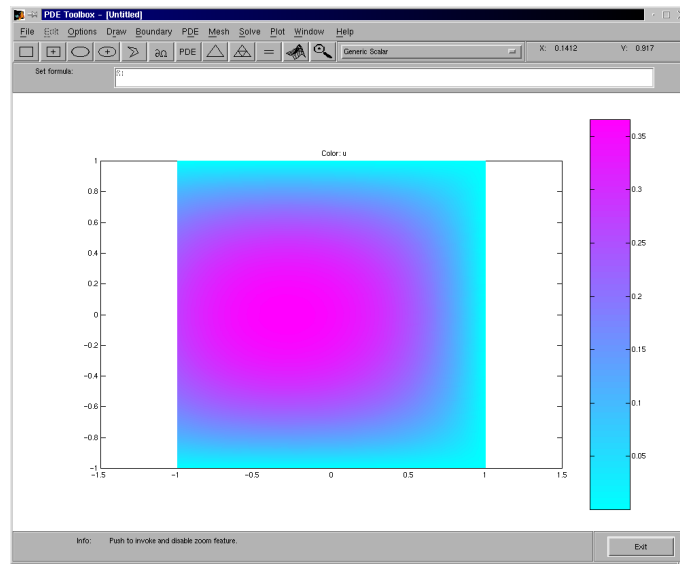


Figura 50: Soluzione

Si può esportare la soluzione nel Workspace come un vettore selezionando l'apposita opzione dal menu *Solve*.

## 9.6 Plot Solution

Possiamo modificare la presentazione del risultato selezionando l'opzione *Plot* e poi *Parameters* da menu oppure cliccando direttamente sull'icona a destra. Verrà



aperta una finestra di dialogo. Per esempio possiamo visualizzare un grafico tridimensionale della soluzione sovrapposto alla mesh tramite le impostazioni riassunte nella Figura 51.

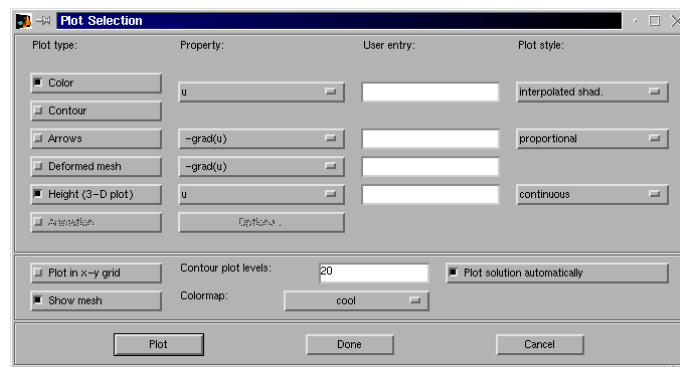


Figura 51: Impostazioni per plot tridimensionale con mesh

Cliccando su *Plot* all'interno della finestra si aprirà una nuova figura che mostriamo in Figura 52.

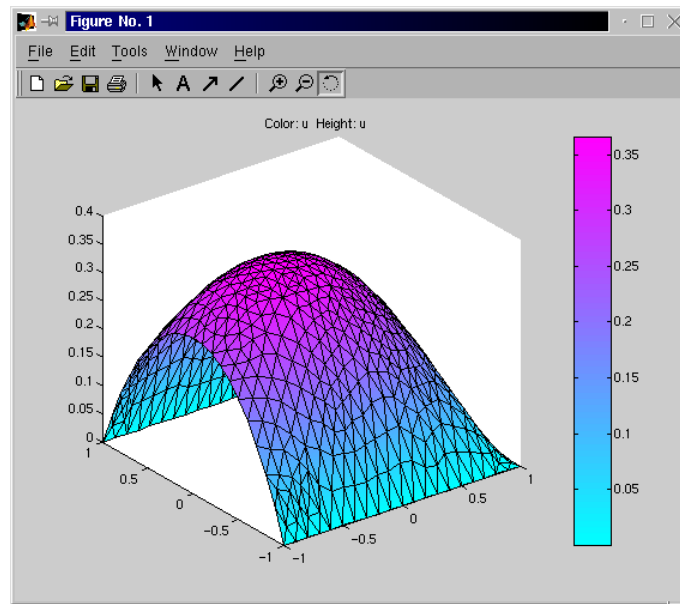


Figura 52: Soluzione con grafico tridimensionale

Possiamo inoltre visualizzare il campo vettoriale  $\nabla u_h$  sovrapposto alle linee di livello tramite le impostazioni che mostriamo in Figura 53.

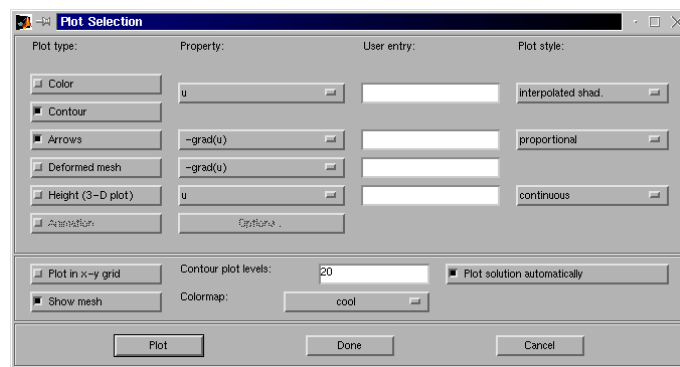


Figura 53: Impostazioni per plot vettoriale e linee di livello

Il risultato è riportato in Figura 54. Il bottone *Done* della finestra di dialogo salva le impostazioni correnti come default.

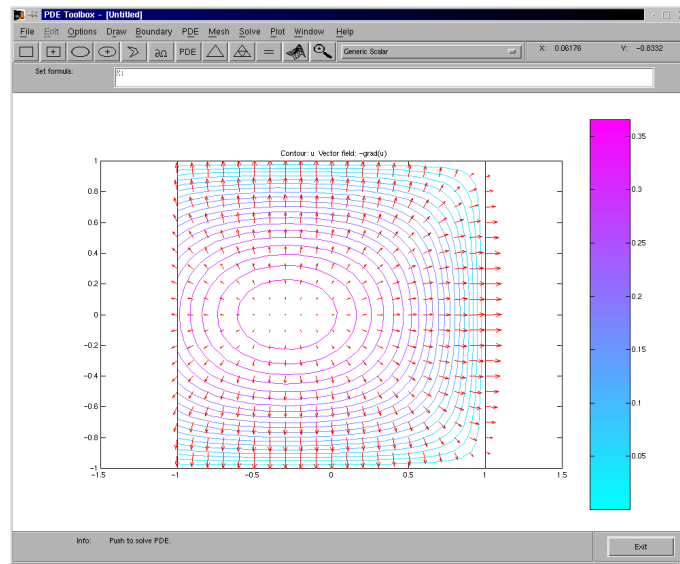


Figura 54: Plot vettoriale e linee di livello

## Riferimenti bibliografici

- [1] M. Bramanti, C.D. Pagani, S. Salsa, *Matematica - Calcolo Infinitesimale e Algebra Lineare*, Zanichelli Editore, 2000.
- [2] W. L. Briggs, V. E. Henson, *The DFT: an owner's manual for the discrete Fourier Transform*, SIAM, Philadelphia, 1995.
- [3] A. Quarteroni, R. Sacco, F. Saleri, *Matematica Numerica*, Springer-Verlag Italia, Milano 2000.
- [4] S. Salsa, A. Squellati, *Esercizi di Analisi Matematica 2 (Parte Terza)*, Ed. Masson, Milano 1994.