

11. EQUAZIONI DIFFERENZIALI ORDINARIE

Consideriamo il seguente problema di Cauchy:

$$\begin{cases} y' = -y + t + 1 \\ y(0) = 1 \end{cases} \quad [11.1]$$

con $t \in [0, T]$.

Il nostro scopo è trovare una soluzione $y(t)$ che soddisfi il problema [11.1].

Prima di effettuare un'integrazione numerica, cerchiamo di ragionare in maniera analitica utilizzando le informazioni a nostra disposizione, forniteci dalla [11.1]. Conosciamo il valore iniziale della nostra soluzione, infatti in $t = 0$ deve essere $y(0) = 1$. Inoltre $y'_{(0,1)} = -y(0) + 0 + 1 = 0$, dunque la funzione nel punto $(t,y)=(0,1)$ avrà un asintoto orizzontale. Procedendo con questo ragionamento, possiamo analizzare l'andamento della funzione $y(t)$ in vari punti, per farci un'idea generale di quale possa essere il suo comportamento qualitativo. Per esempio, in $(t,y)=(1,0)$ si avrà $y'_{(1,0)} = 0 + 1 + 1 = 2$ cioè la funzione è crescente, oppure in $(1,1)$ si avrà $y'_{(1,1)} = -1 + 1 + 1 = 1$ cioè la funzione è ancora crescente.

Estendendo questo concetto, possiamo visualizzare l'andamento della funzione tramite delle frecce poste in un certo numero di punti discreti del piano t - y , che indicano l'andamento di $y(t)$ in quegli specifici punti. Possiamo creare questa griglia in MATLAB usando le seguenti istruzioni:

```
>> t=0:0.1:2;          % discretizzo t in un intervallo [0,2]
>> y=0:0.1:2;          % discretizzo y in un intervallo [0,2]
>> [T,Y]=meshgrid(t,y); % creo i punti [T,Y] e li pongo su una griglia
>> Yp=-Y+T+1;          % definisco y'
>> quiver(T,Y,ones(size(Yp)),Yp) %genero le frecce di flusso
```

Queste istruzioni generano il grafico delle linee di flusso di figura 11-1.

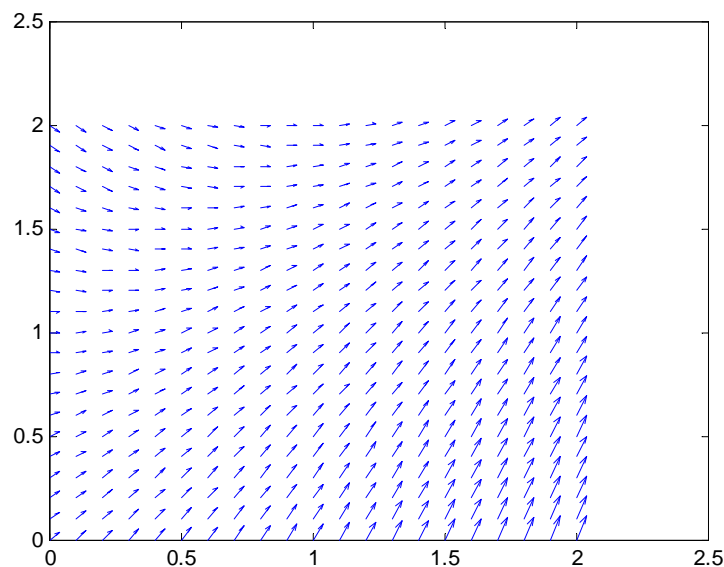


Figura 11-1: Grafico delle linee di flusso della soluzione della [11.1]

Come osserviamo, la soluzione $y(t)$ avrà un andamento differente a seconda del punto iniziale da cui si parte.

In generale, la [11.1] è un'equazione differenziale che è possibile scrivere nella forma generica:

$$\begin{cases} y' = f(t, y(t)) = f(t, y) \\ y(0) = y_0 \end{cases} \quad [11.2]$$

Per esse, vale il seguente:

Teorema-11.1: Se la funzione f è sufficientemente regolare, esiste una ed una sola soluzione del problema.

Definizione-11.1: Una funzione f è detta "sufficientemente regolare" se:

- f è una funzione Lipschitziana in y ;
- f è regolare con decrescenza monotona uniforme.

Definizione-11.2: Una funzione f è detta "Lipschitziana (in un certo intervallo)" se vale:

$$|f(t, y_1) - f(t, y_2)| \leq k \cdot |y_1 - y_2| \quad [11.3]$$

Corollario-11.1: Se f è una funzione regolare con decrescenza monotona uniforme, allora f è Lipschitziana. (Non vale il viceversa).

Nel nostro esercizio, la funzione f è definita come:

$$f(t, y) = -y + t + 1 \quad [11.4]$$

Verifichiamo che sia Lipschitziana e che sia regolare con decrescenza monotona uniforme.

i) *La [11.4] è Lipschitziana?*

Poiché $f(t, y_1) - f(t, y_2) = -y_1 + t + 1 + y_2 - t - 1 = -y_1 + y_2 = -1(y_1 - y_2)$ possiamo concludere che la [11.3] è verificata e dunque la [11.4] è Lipschitziana con costante $k = -1$ (se mettiamo il modulo, allora $k = 1$).

ii) *La [11.4] è regolare con decrescenza monotona uniforme?*

Poiché la derivata parziale vale $f_y = -1$, allora la [11.4] è sempre decrescente e dunque regolare con decrescenza monotona uniforme.

Ora consideriamo un problema differenziale del tipo [11.2] con punto iniziale variato. Il problema differenziale è identico a quello [11.2], l'unica cosa che cambia è il valore iniziale:

$$\begin{cases} y' = f(t, y) \\ y(0) = y_0 \end{cases} \quad \begin{cases} z' = f(t, z) \\ z(0) = z_0 \end{cases} \quad [11.5]$$

La [11.5] presenta uno scenario interessante per fare ragionamenti relativi alla risoluzione numerica di equazioni differenziali. I metodi numerici infatti spesso approssimano le soluzioni e modificano il valore iniziale a causa delle limitazioni di memorizzazione dei numeri macchina. Questo significa che il problema reale è spesso risolto con un problema variato, come nel caso [11.5]; vale dunque la pena cercare di capire quale sia la penalità da pagare quando questo si verifichi.

Per farlo, osserviamo innanzitutto che se f è Lipschitziana, allora vale che:

$$|y(t) - z(t)| \leq e^{k(T-t_0)} \cdot |y_0 - z_0| \quad [11.6]$$

Se f è anche regolare con decrescenza monotona uniforme, vale invece che:

$$|y(t) - z(t)| \leq e^{-(T-t_0)} \cdot |y_0 - z_0| \quad [11.7]$$

Nella condizione [11.6] è presente una forte dipendenza dall'intervallo di integrazione $(T - t_0)$, che accresce esponenzialmente la distanza tra la soluzione del problema originario $y(t)$ e quella del problema variato $z(t)$. Questo significa che con funzioni solo Lipschitziane, la scelta dell'intervallo di integrazione è fondamentale, e solo intervalli piccoli garantiscono una buona vicinanza tra la soluzione reale e quella calcolata dalla macchina.

Nella condizione [11.7], invece, l'esponenziale è negativo e quindi una dimensione grande dell'intervallo di integrazione è auspicabile perché avvicina le soluzioni. Dunque funzioni regolari con decrescenza monotona uniforme sono integrabili con miglior precisione.

Si noti che purtroppo non è possibile agire affatto sulle disequazioni [11.6] e [11.7], che dipendono in maniera quasi esclusiva dalla forma delle funzioni in gioco e che quindi costituiscono un limite teorico all'integrazione numerica.

Metodo di Eulero

Il metodo di Eulero è un popolare metodo di integrazione numerica per funzioni differenziali ordinarie. Esso procede nel calcolo tramite una successione nella forma:

$$\begin{cases} y_{n+1} = y_n + h \cdot f(x_n, y_n) \\ y_0 \end{cases} \quad [11.8]$$

La costante h è detta "passo di discretizzazione" e viene ovviamente scelta a priori.

Usiamo il metodo di Eulero per risolvere la [11.1].

Nel nostro caso, la soluzione iniziale vale $y_0 = 1$.

Prendiamo un intervallo di integrazione $I = \{t \in [0,10]\}$ e fissiamo h , cioè dividiamo l'intervallo I in punti x_0, x_1, \dots equidistanti con distanza h . Cioè:

$$x_{n+1} = x_n + h \quad [11.9]$$

Applicando il metodo [11.8] al problema [11.1] possiamo scrivere che:

$$\begin{cases} y_{n+1} = y_n + h \cdot (-y_n + t_n + 1) \\ y_0 = 1 \end{cases} \quad [11.10]$$

Il metodo risolutivo è già implementato in MATLAB nella funzione `odee`. Per usarla, procediamo come segue.

Per prima cosa scriviamo un m-file che chiameremo `F.m` contenente la funzione [11.4]:

```
% contenuto del file F.m
function dydt=f(t,y)
dydt=-y+t+1;
```

Torniamo ora nella command window di MATLAB e scriviamo il seguente insieme di istruzioni, dove abbiamo scelto un passo di integrazione pari a 0.1:

```
>> h=odeset('InitialStep',0.1);  
>> [t,y]=odee('F',[0,10],1,h);  
>> figure  
>> plot(t,y)
```

Questo genera la soluzione con il metodo di Eulero, che è poi disegnata sul grafico di figura 11-2.

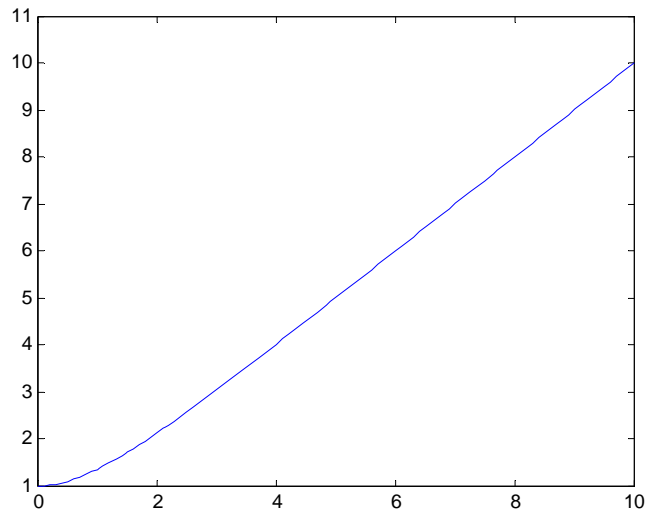


Figura 11-2: Soluzione del problema [11.1] con il metodo di Eulero

Non è difficile osservare che questa soluzione abbia un andamento assolutamente conforme a quello che ci aspettavamo osservando analiticamente il problema [11.1]. Per convincerne ancora di più, effettuiamo un'integrazione simbolica con il comando:

```
>> dsolve('Dy=-y+t+1','y(0)=1')  
  
ans = t+exp(-t)
```

Ci viene restituita una soluzione $t + e^{-t}$ che è proprio quella del grafico.

Calcoliamo ora l'errore tra la soluzione vera e quella numerica:

```
>> yvera=10+exp(-10);  
>> err=abs(yvera-y(end))  
  
err = 1.8839e-005
```

Se vogliamo determinare l'ordine di convergenza p per il metodo di Eulero, dovremmo procedere come nelle precedenti esercitazioni osservando che:

$$E = ch^p \quad [11.11]$$

ed essendo la [11.1] lineare, con diversi passi di discretizzazione si ha che:

$$\begin{aligned} E_1 &= ch_1^p \\ E_2 &= ch_2^p \end{aligned} \quad [11.12]$$

Facendo il rapporto si scrive che:

$$\frac{E_1}{E_2} = \frac{h_1^p}{h_2^p} \quad [11.13]$$

ed infine passando ai logaritmi ed esplicitando per p si giunge a:

$$p = \frac{\log\left(\frac{E_1}{E_2}\right)}{\log\left(\frac{h_1}{h_2}\right)} \quad [11.14]$$

In MATLAB possiamo dunque provare ad integrare con un nuovo passo di integrazione, per esempio 0.01, per poi calcolare la [11.14]:

```
>> h1=odeset('InitialStep',0.01);  
>> [t1,y1]=odee('F',[0,10],1,h1);  
>> err1=abs(yver-y1(end));  
>> log(err/err1)/log(0.1/0.01)
```

MATLAB ci dà come risultato $p = 0,9270$, cioè $p \approx 1$ come era giusto aspettarsi.

Stima dell'errore per il metodo di Eulero

Se f è Lipschitziana con costante k , e $y(t)$ è una funzione $C^2([t_0, T])$, allora vale che:

$$\max|y(t_k) - y_k| \leq c_0 \cdot |y(t_0) - y_0| + c \cdot h \cdot \|y''\|_\infty \quad [11.15]$$

dove $c = \frac{e^{k(T-t_0)} - 1}{2k}$ e $c_0 = e^{k(T-t_0)}$.

$y(t_k)$ e y_k denotano la soluzione vera e quella calcolata da MATLAB rispettivamente.

Se imponiamo che la soluzione approssimata coincida con quella vera nel punto iniziale, allora la [11.15] una volta esplicitate le costanti diventa:

$$\max|y(t_k) - y_k| \leq \frac{e^{k(T-t_0)} - 1}{2k} \cdot h \cdot \|y''\|_\infty \quad [11.16]$$

Dalla [11.6] possiamo determinare h affinché il massimo errore commesso dal metodo di Eulero sia minore o uguale a 10^{-5} .

Sapendo che $k=1$ e che $y'' = e^{-t}$ ha massimo pari ad 1 nell'intervallo $[t_0, T] = [0, 20]$, basta imporre che:

$$10^{-5} \leq \frac{e^{k(T-t_0)} - 1}{2k} \cdot h \cdot \|y''\|_\infty \Rightarrow h \leq \frac{10^{-5} \cdot 2k}{(e^{k(20-0)} - 1) \cdot 1} \approx 4 \cdot 10^{-14} \quad [11.17]$$

Il problema della [11.16] è la necessità di conoscere la derivata seconda y'' , che non sempre è nota o calcolabile.

Possiamo anche calcolare che per $h=0.1$ l'errore nel punto finale $t=10$ vale $E_{t=10} \approx 1.8 \cdot 10^{-5}$. Ci chiediamo dunque quale valore di h sia necessario per ottenere un errore nel punto finale di 10^{-8} . Per fare questo calcolo, si sfrutta la linearità del metodo già osservata in [11.12] e [11.13], la quale ci permette di dire (visto che $p=1$):

$$h_2 = h_1 \frac{E_2}{E_1} = 0.1 \cdot \frac{10^{-8}}{10^{-5}} = 10^{-4}$$

Residuo ed ordine di troncamento unitario

Immaginiamo di arrestare un metodo ad un punto (x_n, y_n) . Supponiamo che $y(x_{n+1})$ sia la soluzione vera e y_{n+1} quella approssimata calcolata dalla macchina. Se partiamo da un punto iniziale $y_n = y(x_n)$ possiamo dare la seguente:

Definizione-11.3: Si definisce residuo di un metodo la differenza:

$$R_n = y(x_{n+1}) - y_{n+1} \quad [11.18]$$

assumendo che $y_n = y(x_n)$.

Per il metodo di Eulero, possiamo calcolare il residuo come segue.

Sappiamo dalla [11.8] che $y_{n+1} = y_n + h \cdot f(x_n, y_n)$, dunque la [11.18] vale:

$$R_n = y(x_{n+1}) - y_{n+1} = y(x_{n+1}) - y_n - h \cdot f(x_n, y_n) \quad [11.19]$$

Tenuto conto della [11.9] e del fatto che $f(x_n, y_n) = y'(x_n)$, la [11.19] diventa:

$$R_n = y(x_n + h) - y_n - h \cdot f(x_n, y_n) \quad [11.20]$$

Sviluppando in serie di Taylor il primo termine si ha che:

$$y(x_n + h) = y(x_n) + hy' + \frac{h^2}{2} y''(\vartheta) \quad [11.21]$$

E allora, sostituendo la [11.21] nella [11.20] e tenuto conto dell'assunzione $y_n = y(x_n)$, il residuo per il metodo di Eulero vale infine:

$$R_n = y(x_n) + hy' + \frac{h^2}{2} y''(\vartheta) - y_n - h \cdot f(x_n, y_n) = \frac{h^2}{2} y''(\vartheta) \quad [11.22]$$

Grazie al residuo possiamo anche definire un'altra importante grandezza, data dalla seguente:

Definizione-11.4: L'errore di troncamento locale unitario è dato dal rapporto:

$$\tau = \frac{R_n}{h} \quad [11.23]$$

Per Eulero, esso vale:

$$\tau_{Eulero} = \frac{h}{2} y''(\vartheta) \quad [11.24]$$

Per tutti i tipi di metodi si ha sempre che il massimo errore è proporzionale a τ .