Metodi Numerici con Laboratorio di Informatica - A.A. 2015-2016
Esercizi Laboratorio n° 4 - Metodo di Newton e Metodi di punto fisso

Metodi numerici per le equazioni differenziali ordinarie

Consideriamo un generico problema di Cauchy:
trovare una funzione $y : I \longrightarrow \mathrm{R}$, con $y \in \mathcal{C}^1(I)$, tale che

$$\begin{cases} y'(t) = f(t, y) \quad t \in I \\ y(t_0) = y_0 \end{cases} \tag{1}$$

dove $I := [t_0, t_{max}]$, $y_0 \in \mathrm{R}$ e $f : I \times \mathrm{R} \longrightarrow \mathrm{R}$ è una funzione assegnata.
I metodi numerici utilizzati per risolvere il problema (1) si basano sulla seguente strategia:

1. stabilire un passo di avanzamento temporale h;
2. suddividere l'intervallo temporale $[t_0, t_{max}]$ in un numero N_h di sottointervalli di ampiezza h
$$N_h = (t_{max} - t_0)/h;$$
3. per ogni istante temporale discreto t_n, con $t_0 < t_n \leq t_{max}$, calcolare il valore u_n che approssima la soluzione di (1) $y(t_n)$.

L'insieme dei valori $\{u_0 = y_0, u_1, \ldots, u_{N_h}\}$ rappresenta la soluzione numerica di (1).

- Il **metodo di Eulero esplicito** calcola la soluzione numerica $\{u_n\}$ di (1) generando la successione

$$\boxed{u_0 = y_0, \quad u_{n+1} = u_n + hf(t_n, u_n), \quad n = 0, \ldots, N_h - 1.}$$

- Il **metodo di Eulero implicito** calcola la soluzione numerica $\{u_n\}$ di (1) generando la succesione di valori

$$\boxed{u_0 = y_0, \quad u_{n+1} = u_n + hf(t_{n+1}, u_{n+1}), \quad n = 0, \ldots, N_h - 1.}$$

  Tale metodo è implicito e ad un passo, in quanto ad ogni passo temporale la soluzione numerica u_{n+1} dipende da u_{n+1}, che è incognita, e da u_n che è stata calcolata al passo precedente. Quindi se f è una funzione non lineare ad ogni istante temporale si deve risolvere un'equazione non lineare nell'icognita u_{n+1} utilizzando, ad esempio, il metodo di Newton. Infatti, u_{n+1} è lo zero della funzione

$$\varphi(w) \equiv u_n + hf(t_{n+1}, w) - w.$$

  Il metodo di Newton per questa equazione si scrive

$$\begin{cases} w^{(0)} = u_n \\ w^{(k+1)} = w^{(k)} - \dfrac{\varphi(w^{(k)})}{\varphi'(w^{(k)})} \end{cases}$$

  dove $\varphi'(w) = h\frac{\partial f}{\partial y}(t_{n+1}, w) - 1$. Per costruire le iterate del metodo di Newton, sarà necessario fornire in input alla funzione l'espressione di $\frac{\partial f}{\partial y}(t, y)$.

- Il metod**o di Crank-Nicolson** calcola la soluzione numerica $\{u_n\}$ di (1) generando la successione di valori

$$\boxed{u_0 = y_0, \quad u_{n+1} = u_n + \frac{h}{2}[f(t_n, u_n) + f(t_{n+1}, u_{n+1})], \quad n = 0, \ldots, N_h - 1,}$$

  È un metodo implicito e, analogamente al metodo di Eulero implicito, ad ogni istante temporale si deve risolvere un'equazione non lineare utilizzando il metodo di Newton.

Esercizio 1. Si consideri il problema di Cauchy

$$\begin{cases} y'(t) = \cos(t)\exp(-t/2) - \frac{1}{2}y \quad 0 < t \leq 10 \\ y(0) = 0 \end{cases} \tag{2}$$

La soluzione esatta di tale problema, nell'intervallo limitato $t \in [0, 10]$ è

$$y(t) = \sin(t)\exp(-t/2).$$

1. Rappresentare graficamente la soluzione esatta nell'intervallo considerato.

2. Implementare i metodi di Eulero Esplicito, Eulero Implicito e Crank-Nicolson nelle function Matlab di cui si riporta l'intestazione

```
function [T,u] = eulero_esplicito(f,tspan,y0,h)
function [T,u] = eulero_implicito(f,dfy,tspan,y0,h)
function [T,u] = crank_nicolson(f,dfy,tspan,y0,h)
```

   dove

   - `f` è la funzione del problema di cauchy (`f = @(t,y) ...`);
   - `dfy` è la derivata di f rispetto a y (`df = @(t,y) ...`);
   - `tspan=[t0,t_max]` è il vettore contenente gli estremi dell'intervallo temporale
   - `y0` è il dato iniziale;
   - `h` è il passo temporale;
   - `T` è vettore degli istanti in cui si calcola la soluzione discreta;
   - `u` è il vettore contenente l'approssimazione ad ogni istante temporale.

3. Risolvere numericamente il problema (2) utilizzando le function implementate ponendo $h = 0.5$.
   Rappresentare sullo stesso grafico le soluzioni numeriche ottenute e confrontarle con la soluzione esatta.

4. Risolvere numericamente il problema con i tre metodi utilizzando i passi di discretizzazione temporale: $h = [0.4,\ 0.2,\ 0.1,\ 0.05,\ 0.025,\ 0.0125]$. Al variare di h, si valuti per ogni metodo il massimo modulo dell'errore compiuto approssimando la soluzione esatta $y(t_n)$ con la soluzione numerica u_n:

$$e_h = \max_{t_n \in [t_0, t_{max}]} |y(t_n) - u_n|,$$

5. Riportare, su un grafico in scala logaritmica su entrambi gli assi, l'andamento di e_h al variare di h per i tre metodi considerati. Verificare che ci sia accordo con gli ordini di convergenza teorici dei metodi.

Traccia della soluzione

1. Assegnare le funzioni `f`, `dfy` e la soluzione esatta `y_ex`, gli estremi dell'intervallo `tspan=[t0 t_max]`, il dato iniziale `y0`.
2. Creare un vettore dei tempi `t_plot` su cui valutare la soluzione esatta per disegnarne il grafico.
3. Assegnare il vettore `H=[0.4 0.2 0.1 0.05 0.025 0.0125]`.
4. Per ciascun valore di `H` (`for i=1:length(H)`)
   - calcolare la soluzione numerica con ciascun metodo `u_ee`, `u_ei` e `u_ei`
   - disegnare il grafico
     ```
     plot(T,u_ee,'bo-',T,u_ei,'ro-',T,u_cn,'go-',t_plot,y_ex(t_plot),'k');
     ```
   - calcolare l'errore
     ```
     E1(i)=max(abs(u_ee-y_ex(T))); E2(i)=max(abs(u_ei-y_ex(T)));
     E3(i)=max(abs(u_cn-y_ex(T)));
     ```
5. Plottare gli errori in scala logaritmica confrontandoli con le stime teoriche:
   ```
   loglog(H,E1,'o-',H,E2,'o-',H,E3,'o-',H,H,'k',H,H.^2,'k--')
   legend('Eulero avanti','Eulero indietro', 'Crank Nicolson', 'h', 'h^2')
   ```

Sistemi di equazioni differenziali ordinarie

Si consideri il seguente sistema di m equazioni differenziali ordinarie nelle incognite $y_1 = y_1(t), \ldots, y_m = y_m(t)$:

$$\begin{cases} y_1' = f_1(t, y_1, \ldots, y_m) \\ \quad \vdots \\ y_m' = f_m(t, y_1, \ldots, y_m) \end{cases}$$

dove $t \in (t_0, T]$, corredato dalle condizioni iniziali

$$y_1(t_0) = y_{0,1}, \ \ldots \ y_m(t_0) = y_{0,m}.$$

Matlab fornisce delle funzioni predefinite per la risoluzione accurata di questi problemi tra cui `ode45` e `ode23` per problemi *non-stiff* e `ode15s` e `ode23s` per problemi *stiff*. La sintassi di queste function è

```
[T,y] = odeXX(f,tspan,y0)
```

dove

- `f` è la funzione del problema di cauchy (`f = @(t,y) [...; ...;..]`);
- `tspan=[t0,t_max]` è il vettore contenente gli estremi dell'intervallo temporale
- `y0` è il vettore colonna contenente i dati iniziali;
- `T` è vettore degli istanti in cui si calcola la soluzione discreta;
- `y` è la matrice `lenght(T)` $\times m$ contenente la soluzione numerica: la riga $i-$esima corrisponde alla soluzione al tempo `T(i)`.

Esempio 1: Modello Brusselator
Il modello Brusselator descrive una reazione autocatalitica e oscillante. Supponiamo che sei sostanze (A, B, D, E, X e Y) interagiscano secondo il seguente schema

$$\begin{aligned} \mathrm{A} &\xrightarrow{k_1} \mathrm{X} \\ \mathrm{B} + \mathrm{X} &\xrightarrow{k_2} \mathrm{Y} + \mathrm{D} \qquad \text{reazione bimolecolare} \\ 2\mathrm{X} + \mathrm{Y} &\xrightarrow{k_3} 3\mathrm{X} \qquad \text{reazione trimolecolare autocatalitica} \\ \mathrm{X} &\xrightarrow{k_4} \mathrm{E} \end{aligned}$$

Se denotiamo con $A(t)$, $B(t)$,...etc. le concentrazioni di A, B...etc in funzione del tempo, utilizzando la *legge di azione di massa* possiamo descrivere la dinamica delle sei sostanze secondo il seguente sistema di equazioni differenziali

$$\begin{aligned} A' &= -k_1 A \\ B' &= -k_2 BX \\ D' &= k_2 BX \\ E' &= k_4 X \\ X' &= k_1 A - k_2 BX + k_3(3-2)X^2 Y - k_4 X \\ Y' &= k_2 BX - k_3 X^2 Y. \end{aligned}$$

Questo sistema si può semplificare eliminando le equazioni per D e E, in quanto non influenzano le altre, e assumendo che le concentrazioni di A e B siano costanti nel tempo, cioè $A(t) \equiv a$ e $B(t) \equiv b$. Inoltre ponendo le velocità di reazione $k_i = 1$ e rinominando $y_1(t) := X(t)$ e $y_2(t) := Y(t)$ otteniamo il seguente sistema a due equazioni

$$\boxed{\begin{aligned} {y_1}' &= a - by_1 + {y_1}^2 y_2 - y_1 \\ {y_2}' &= by_1 - {y_1}^2 y_2. \end{aligned}}$$

Il seguente codice Matlab risolve il sistema Brusselator utilizzando `ode45` e traccia il grafico delle soluzioni e del ritratto di fase per $a = 1$ e $b = 3$.

```
%Brusselator
clear all; close all
a=1;
b=3;
f = @(t,y)[ a + y(1).*(y(1).*y(2)-(b+1)); y(1).*(b-y(1).*y(2))];
[t,y] = ode45(f,[0 20],[1.5;3],options);
%% grafico delle soluzioni
figure(1)
plot(t,y(:,1),'r',t,y(:,2),'b')
xlabel('t')
legend('y_1','y_2')
%% ritratto di fase
figure(2)
plot(y(:,1),y(:,2))
xlabel('y_1')
ylabel('y_2')
```

Esempio 2: Problema stiff

Un tipico esempio di problema *stiff* è il sistema di equazioni differenziali ordinarie che modellizza la seguente reazione autocatalitica (detta reazione di Robertson)

$$\begin{aligned} \mathrm{A} &\xrightarrow{k_1} \mathrm{B} \qquad k_1 = 0.04 \\ \mathrm{B} + \mathrm{B} &\xrightarrow{k_2} \mathrm{C} + \mathrm{B} \qquad k_2 = 10^7 \\ \mathrm{B} + \mathrm{C} &\xrightarrow{k_3} \mathrm{A} + \mathrm{C} \qquad k_3 = 10^4 \end{aligned}$$

Ponendo $y_1(t) = A(t)$, $y_2(t) = B(t)$ e $y_3(t) = C(t)$ e utilizzando la *legge di azione di massa* possiamo descrivere la dinamica delle tre sostanze secondo il seguente sistema di equazioni differenziali

$$\boxed{\begin{aligned} {y_1}' &= -k_1 y_1 + k_3 y_2 y_3 \\ {y_2}' &= k_1 y_1 - k_2 {y_2}^2 - k_3 y_2 y_3 \\ {y_3}' &= k_2 {y_2}^2. \end{aligned}}$$

Il seguente codice Matlab risolve il sistema di Robertson utilizzando `ode45` e `ode15s` e calcola i tempi di esecuzione per entrambi i metodi. In questo particolare esempio si può osservare che `ode15s` è circa 2000 volte più veloce rispetto a `ode45`.

```
%Robertson reaction
clear all;close all
k1=0.04; k2=3e7; k3=1e4;
f =@(t,y) [−k1*y(1) + k3*y(2)*y(3);
           k1*y(1)− k2*y(2)^2 − k3*y(2)*y(3);
           k2*y(2)^2];
tic
[t,y] = ode45(f,[0 100],[1;0;0]);
toc
plot(t,y(:,1),t,y(:,2), t, y(:,3))
xlabel('t')
legend('y_1','y_2','y_3')

tic
[ts,ys] = ode15s(f,[0 100],[1;0;0]);
toc
figure
plot(ts,ys(:,1),ts,ys(:,2), ts, ys(:,3))
xlabel('t')
legend('y_1','y_2','y_3')
```