Serie 7
Equazioni Differenziali Ordinarie

1 Approssimazione delle derivate di una funzione

Consideriamo il problema dell'approssimazione delle derivate di una funzione f in un punto $\overline{x} \in [a, b]$. Supponiamo che la funzione f sia continua e derivabile in un intorno del punto $\overline{x} \in [a, b]$, allora

$$f'(\overline{x}) = \lim_{h \to 0} \frac{f(\overline{x} + h) - f(\overline{x})}{h}.$$

L'idea dei metodi alle differenze finite è quella di sostituire all'operazione di limite il rapporto incrementale con h finito. In questo modo, si ottengono i tre metodi per l'approssimazione di $f'(\overline{x})$:

- Differenze finite in avanti : $f'(\overline{x}) \approx \delta_+ f(\overline{x}) = \dfrac{f(\overline{x} + h) - f(\overline{x})}{h}$,
- Differenze finite all'indietro : $f'(\overline{x}) \approx \delta_- f(\overline{x}) = \dfrac{f(\overline{x}) - f(\overline{x} - h)}{h}$,
- Differenze centrate: $f'(\overline{x}) \approx \delta_c f(\overline{x}) = \dfrac{f(\overline{x} + h) - f(\overline{x} - h)}{2h}$.

L'approssimazione della derivata seconda $f''(\overline{x})$ tramite le differenze finite centrate è:

$$f''(\overline{x}) \approx \delta_c^2 f(\overline{x}) = \frac{f(\overline{x} + h) - 2f(\overline{x}) + f(\overline{x} - h)}{h^2}.$$

Esercizio 1.1

Si vogliono approssimare le derivate prima e seconda della funzione

$$f(x) = e^{-x^2} \sin(2x + 1) \qquad \text{in } \overline{x} = 0.$$

1. Si approssimi $f'(0)$ tramite gli schemi alle differenze finite in avanti, all'indietro e centrate, utilizzando un passo $h = 0.4, 0.2, 0.1, 0.05, 0.025, 0.0125$. Per ciascuno dei tre schemi si calcoli l'errore commesso e se ne visualizzi l'andamento in funzione del passo h su un grafico in scala logaritmica su entrambu gli assi; verificare che ci sia accordo con i risultati teorici.

2. Si ripeta il punto 1 per l'approssimazione di $f''(0)$ tramite lo schema delle differenze finite centrate.

2 Metodi numerici per l'approssimazione di equazioni differenziali ordinarie (EDO)

Dato un generico problema di Cauchy:

$$\begin{cases} y'(t) = f(t, y(t)) & t_0 < t < t_{max} \\ y(t_0) = y_0 \end{cases} \tag{1}$$

i metodi numerici utilizzati per risolvere il problema (4) 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

   $$N_h = (t_{max} - t_0)/h$$

   di egual ampiezza $h > 0$;
3. per ogni istante temporale discreto t_n, con $t_0 < t_n \leq t_{max}$, si calcola il valore incognito u_n che approssima la soluzione di (4) $y_n = y(t_n)$.

L'insieme dei valori $\{u_0 = y_0, u_1, \ldots, u_{N_h}\}$ rappresenta la soluzione numerica di (4).

Metodi di Eulero in avanti e all'indietro

Il metodo di Eulero in avanti calcola la soluzione numerica $\{u_n\}$ di (4) generando la successione di valori:

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

Osserviamo che tale metodo è esplicito e ad un passo (one-step) in quanto, ad ogni passo temporale, la soluzione numerica u_{n+1} dipende soltanto dalla soluzione al passo temporale precedente u_n.

Il metodo di Eulero all'indietro calcola invece la soluzione numerica $\{u_n\}$ di (4) generando la successione di valori:

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

Tale metodo è implicito e ad un passo (one-step), in quanto, ad ogni passo temporale, la soluzione numerica u_{n+1} dipende dalla stessa soluzione incognita u_{n+1}, oltre che da u_n. Quindi se f è una funzione non lineare, ad ogni istante temporale, si deve risolvere un'equazione non lineare nell'incognita u_{n+1} utilizzando, ad esempio il metodo di Newton o il metodo delle iterazioni di punto fisso.

Si osservi che l'utilizzo di un metodo delle iterazioni punto fisso richiede che il modulo della derivata della funzione di iterazione $\phi_n(y) = u_n + h\, f(t_{n+1}, y)$ utilizzata sia minore di uno per $y = u_{n+1}$ per ogni $n = 0, 1, \ldots, N_h - 1$ ($|\phi_n'(u_{n+1})| < 1$ per ogni $n = 0, 1, \ldots, N_h - 1$); questa condizione è generalmente soddisfatta per h sufficientemente piccolo.

Esercizio 2.1

Si consideri il problema di Cauchy (4) con la seguente funzione

$$f(t, y) = \lambda\, y, \qquad \text{con } \lambda \text{ un numero reale poisitivo.} \tag{2}$$

Si osservi che questa scelta specifica non abbiamo il problema modello, perché in questo caso $\lambda > 0$. La soluzione esatta di tale problema, nell'intervallo limitato $t \in [t_0, t_{max}]$ è

$$y(t) = y_0\, e^{\lambda\,(t-t_0)}, \qquad t \geq t_0.$$

1. Scrivere la funzione Matlab® `eulero_avanti.m` che implementa il metodo di Eulero in avanti per la risoluzione di un generico problema di Cauchy (4).

   La funzione `eulero_avanti.m` riceve in ingresso:

   - la funzione $f(t, y)$ definita come `inline` o anonymous function (tramite il comando `@`). Si noti che f è una funzione di due variabili;
   - l'estremo t_{max} dell'intervallo di definizione;
   - il dato iniziale y_0;
   - il passo di avanzamento temporale h.

   La funzione `eulero_avanti.m` restituisce il vettore `t_h` degli istanti temporali discreti e il vettore `u_h` dei valori della corrispondente soluzione approssimata.

   L'intestazione di `eulero_avanti.m` sarà quindi:

   ```
   function [t_h,u_h]=eulero_avanti(f,t_max,y_0,h)
   ```

2. Utilizzando la funzione appena scritta al punto 1, risolvere il problema di Cauchy (4) con la funzione $f(t, y)$ (2) con il metodo di Eulero in avanti, scegliendo $\lambda = 2.4$, $t_0 = 0$, $t_{max} = 3$, $y_0 = 0.1$ per i valori del passo temporale $h_1 = 0.05$ e $h_2 = 0.01$. Rappresentare sullo stesso grafico le soluzioni numeriche ottenute e confrontarle con la soluzione esatta.

3. Scrivere la funzione Matlab® `eulero_indietro.m` che implementa il metodo di Eulero all'indietro per la risoluzione di un generico problema di Cauchy (4).

   La funzione `eulero_indietro.m` riceve in ingresso:

   - la funzione $f(t, y)$ definita come `inline` o anonymous function (tramite il comando `@`). Si noti che f è una funzione di due variabili;
   - l'estremo t_{max} dell'intervallo di definizione;
   - il dato iniziale y_0;
   - il passo di avanzamento temporale h.

   La function `eulero_indietro.m` restituisce il vettore `t_h` degli istanti temporali discreti e il vettore `u_h` dei valori della corrispondente soluzione approssimata. Inoltre fornisce in uscita il vettore `iter_pf` che contiene il numero delle iterazioni effettuate ad ogni passo temporale, dal metodo delle *iterazioni di punto fisso*, qui utilizzato per risolvere l'equazione non lineare corrispondente.

L'intestazione di `eulero_indietro.m` sarà quindi:

```
function [t_h,u_h,iter_pf]=eulero_indietro(f,t_max,y_0,h);
```

Utilizzare la function `ptofisso.m` nell'implementazione di `eulero_indietro.m` per risolvere l'equazione non lineare ad ogni passo temporale del metodo di Eulero all'indietro. Si scelgano, per la function `ptofisso.m` una tolleranza pari a 10^{-5} e un numero massimo di iterazioni pari a 100.

4. Utilizzando la funzione appena scritta al punto 3, risolvere il problema (2) con il metodo di Eulero all'indietro, scegliendo $\lambda = 2.4$, $t_0 = 0$, $t_{max} = 3$, $y_0 = 0.1$ per i valori del passo $h_1 = 0.05$ e $h_2 = 0.01$. Rappresentare sullo stesso grafico le soluzioni numeriche ottenute e confrontarle con la soluzione esatta.

5. Riportare su un grafico l'andamento del numero di iterazioni impiegate dal metodo delle iterazioni di punto fisso per risolvere l'equazione non lineare con il metodo di Eulero all'indietro in funzione degli istanti temporali, scegliendo $\lambda = 2.4$, $t_0 = 0$, $t_{max} = 3$, $y_0 = 0.1$ e $h_1 = 0.05$.

6. Definito $e_h = \max_{n=0,1,\ldots,N_h} |y(t_n) - u_n|$, il massimo modulo dell'errore compiuto approssimando la soluzione esatta $y(t_n)$ con la soluzione numerica u_n, riportare, su un grafico in scala logaritmica, l'andamento di e_h al variare di h, utilizzando i passi temporali $h = 0.04, 0.02, 0.01, 0.005, 0.0025$ per entrambi i metodi di Eulero in avanti e all'indietro. Si commenti il risultato ottenuto alla luce della teoria.

Esercizio 2.2

Si ripeta l'Esercizio 2 con

$$f(t,y) = \left[\pi \frac{\cos(\pi t)}{2 + \sin(\pi\, t)} - \frac{1}{2}\right] y,$$

$y_0 = 2$, $t_0 = 0$, $t_{max} = 10$ e sapendo che

$$y(t) = (2 + \sin(\pi\, t))\; e^{-t/2}, \qquad t \geq t_0 = 0.$$

Regione di assoluta stabilità per il metodi di Eulero in avanti e all'indietro

Si consideri il seguente problema modello:

$$\begin{cases} y'(t) = \lambda y(t) & t > t_0 \\ y(t_0) = y_0 \end{cases} \tag{3}$$

con $Re(\lambda) < 0$. La soluzione esatta è la funzione esponenziale decrescente:

$$y(t) = y_0\, e^{\lambda(t-t_0)} \qquad \text{per } t \geq t_0.$$

Sia u_n la soluzione numerica del problema (3): definiamo *regione di assoluta stabilità* del metodo numerico utilizzato, l'insieme dei valori $z = h\lambda$ nel piano complesso $\mathbb{C}$ tali per cui

$$\lim_{n\to\infty} u_n = 0,$$

ovvero tale per cui la funzione di stabilità $|R(\lambda h)| < 1$, essendo $u_n = (R(\lambda h))^n\, y_0$ per $n = 0, 1 \ldots, n$. La regione di assoluta stabilità del metodo di Eulero in avanti è

$$\mathcal{E}_a = \{z = \lambda h \in \mathbb{C},\ \text{t.c. } |z+1| < 1\},$$

essendo $R(z) = (1+z)$; invece la regione di assoluta stabilità del metodo di Eulero all'indietro è

$$\mathcal{E}_i = \{z = \lambda h \in \mathbb{C},\ \text{t.c. } |z-1| > 1\},$$

essendo $R(z) = \frac{1}{1-z}$ per tale metodo. La Figura 1 mostra le regioni di assoluta stabilità dei metodi di Eulero in avanti e all'indietro.

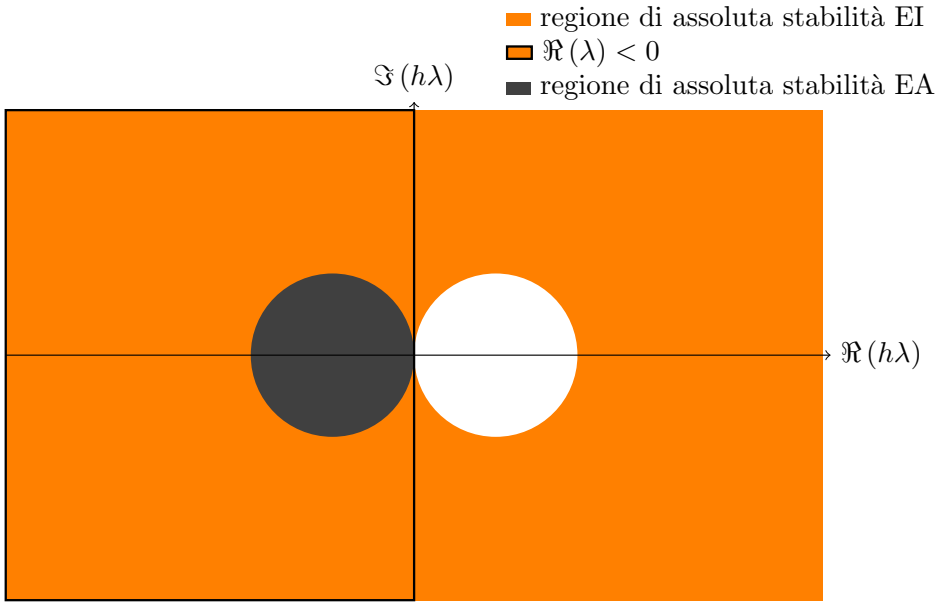

Figura 1: Regioni di assoluta stabilità del metodo di Eulero in avanti e all'indietro. Il rettangolo nero delimita la regione del piano complesso $Re(\lambda h) < 0$, relativa al problema modello (3).

Si osservi che se il problema modello viene definito per $Re(\lambda) < 0$, si devono considerare solo le regioni di assoluta stabilità appartenenti al semipiano complesso $Re(\lambda h) < 0$.

Esercizio 2.3

Si consideri il problema modello (3), con $\lambda = -42$, $y_0 = 2$ e $t_0 = 0$, la cui soluzione esatta è:

$$y(t) = y_0\, e^{\lambda t} = 2\, e^{-42t} \qquad \text{per } t \geq 0.$$

1. Riportare il grafico della soluzione esatta $y(t)$.
2. Usando la funzione `eulero_avanti`, calcolare la soluzione numerica del problema (3) utilizzando il metodo di Eulero in avanti con i seguenti dati: $t_{max} = 1$, e $h = 0.05$. Visualizzare sullo stesso grafico la soluzione esatta e la soluzione numerica. Ripetere

la stessa procedura utilizzando il metodo di Eulero all'indietro con metodo di Newton per la soluzione dell'equazione non lineare di avanzamento temporale (funzione `eulero_indietro_newton`). Cosa si osserva? Valutare se il valore λh appartiente alla regione di assoluta stabilità dei due metodi.

3. Ripetere la stessa analisi del punto precedente usando $h = 0.01$.

4. (*Importanza del solutore non lineare*) La scelta del metodo utilizzato per la risoluzione dell'equazione (in genere non lineare) associata ad ogni passo di un metodo implicito può modificare le proprietà di convergenza generali appena viste. L'utilizzo ad esempio di un metodo delle iterazioni di punto fisso nell'implementazione del metodo di Eulero all'indietro introduce un'ulteriore richiesta che il metodo numerico deve soddisfare. Infatti, l'approssimazione numerica u_{n+1} della soluzione al passo t_{n+1} viene calcolata risolvendo il problema di punto fisso associato alla funzione di iterazione:

$$\phi(x) = u_n + \lambda h x$$

dove u_n è l'approssimazione numerica della soluzione al passo t_n per il problema modello. Pertanto, la condizione $|\phi'(u_{n+1})| < 1$ del Teorema di Ostrowski, si traduce in un vincolo sul passo di discretizzazione h:

$$|\lambda h| < 1.$$

La circonferenza nel piano complesso $\lambda h = 1$ è rappresentata (sovrapposta alle regioni di stabilità) in Figura 2 in colore blu.

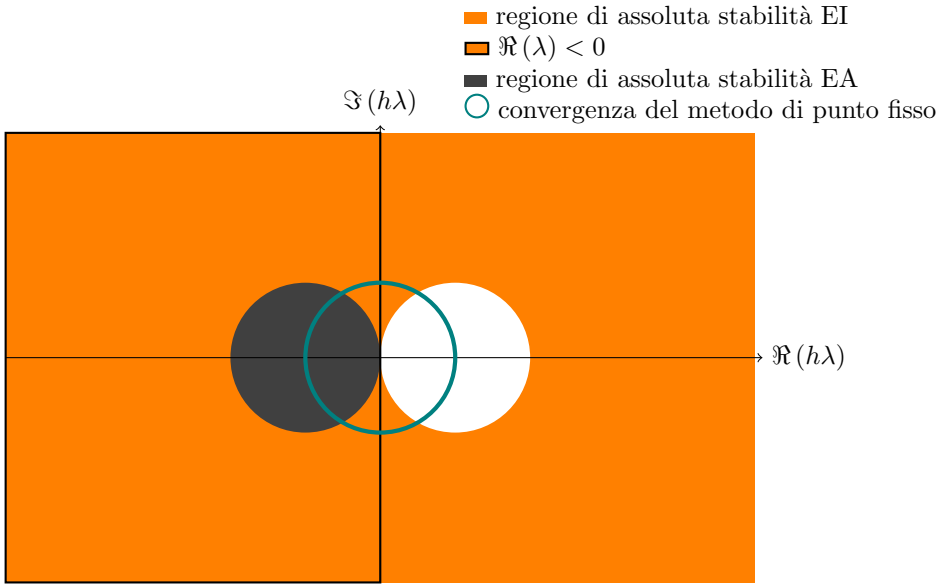

Figura 2: Regioni di assoluta stabilità del metodo di Eulero in avanti e all'indietro. Il rettangolo nero delimita la regione del piano complesso $Re(\lambda h) < 0$, relativa al problema modello (3). La circonferenza blu delimita la regione di convergenza del metodo di punto fisso.

(a) Calcolare la soluzione numerica del problema (3) utilizzando il metodo di Eulero in avanti con i seguenti dati: $t_{max} = 1$, e $h = 0.05$. Visualizzare sullo stesso grafico la soluzione esatta e la soluzione numerica. Ripetere la stessa procedura utilizzando il metodo di Eulero all'indietro con metodo di punto fisso per la soluzione dell'equazione non lineare di avanzamento temporale (funzione `eulero_indietro_pto_fisso`). Cosa si osserva? Valutare se il valore λh appartiene alla regione di assoluta stabilità dei due metodi.

(b) Ripetere la stessa analisi del punto precedente usando prima $h = 0.03$ e poi $h = 0.01$

Metodi di Crank-Nicolson e Heun

Dato un generico problema di Cauchy:

$$\begin{cases} y'(t) = f(t, y) \quad t_0 < t \leq t_{max} \\ y(t_0) = y_0. \end{cases} \tag{4}$$

Il metodo di Crank-Nicolson calcola la soluzione numerica $\{u_n\}$ di (4) generando la successione di valori:

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

dove h è il passo temporale di discretizzazione e $N_h = (t_{max} - t_0)/h$ è il numero di intervalli temporali utilizzati. Osserviamo che tale metodo è implicito in quanto, ad ogni passo temporale la soluzione numerica u_{n+1} dipende dalla stessa soluzione incognita u_{n+1}. Analogamente a quanto visto per il metodo di Eulero all'indietro, ad ogni istante temporale si deve risolvere un'equazione non lineare utilizzando, ad esempio, un metodo delle iterazioni di punto fisso.

Il metodo di Heun calcola la soluzione numerica $\{u_n\}$ di (4) generando la successione di valori:

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

si tratta di un metodo esplicito.

Metodi di Runge-Kutta in Matlab®

Matlab® fornisce delle funzioni built-in che permettono di risolvere EDO utilizzando metodi di Runge-Kutta. La sintassi da utilizzare per chiamare queste funzioni è

```
[t,y] = odeXY(fun,[t0 tf],y0)
```

dove `y0` è la condizione iniziale, `[t0 tf]` è l'intervallo temporale all'interno del quale si vuole risolvere il problema e `fun` è una funzione specificata dall'utente. `odeXY` è uno dei metodi disponibili in Matlab® (ad esempio, `ode23` o `ode45`; si veda la documentazione dei metodi per approfondimenti). L'output della funzione è composto dal vettore `t`, che contiene gli istanti temporali di discretizzazione, e dal vettore `y` che contiene i valori assunti dalla funzione approssimante negli istanti temporali `t`. L'utente deve quindi implementare la *anonymous function* `fun` che restituisce la valutazione dei termini di destra del sistema per un dato istante.

Ad esempio, se si volesse risolvere un problema di Cauchy con $f(t, y) = -\dfrac{1}{2 + \sin(t)}\, y$ per $t \in [0, 10)$ e $y_0 = 7$, i comandi da utilizzare sarebbero:

```
>> y0=7;
>> tmax=10;
>> f = @( t, y ) - 1 ./ ( 2 + sin( t ) ) .* y;
>> [t_h, u_h] = ode45 (f, [0, tmax], y0);
```

Il passo temporale di discretizzazione non deve essere fornito in quanto viene scelto adattivamente da Matlab® per garantire errore al di sotto di una certa tolleranza che può essere fissata dall'utente (si vedano l'`help` e la documentazione per maggiori dettagli).

Esercizio 2.4

1. Scrivere la function `Crank_Nicolson.m` che implementa il metodo di Crank-Nicolson per la risoluzione del problema di Cauchy (4) utilizzando la function `ptofis.m` per risolvere l'equazione non lineare ad ogni passo temporale del metodo. Si scelgano, per la function `ptofis.m` una tolleranza pari a 10^{-5} e un numero massimo di iterazioni pari a 100.
   La funzione `Crank_Nicolson.m` riceve in ingresso:

   - la funzione $f(t, y)$ definita come anonymous function (tramite il comando `@`). Si noti che f è una funzione di due variabili;
   - l'estremo t_{max} dell'intervallo di definizione;
   - il dato iniziale y_0;
   - il passo di avanzamento temporale h.

   Si suppone che l'istante iniziale sia $t_0 = 0$. La funzione `Crank_Nicolson.m` restituisce il vettore `t_h` degli istanti temporali discreti e il vettore `u_h` dei valori della corrispondente soluzione approssimata. Inoltre fornisce in uscita il vettore `iter_pf` che contiene il numero delle iterazioni effettuate dal metodo di punto fisso ad ogni passo temporale per risolvere l'equazione non lineare.

   L'intestazione di `Crank_Nicolson.m` sarà quindi:

   ```
   function [t_h,u_h,iter_pf]=Crank_Nicolson(f,t_max,y_0,h)
   ```

2. Utilizzando la function appena scritta, risolvere il problema (3) con il metodo di Crank-Nicolson, scegliendo $\lambda = -42$, $t_{max} = 1$, $y_0 = 2$ per il valore del passo $h = 0.02$.

3. Riportare su un grafico il numero di iterazioni impiegate dal metodo di punto fisso per risolvere l'equazione non lineare ad ogni passo temporale del metodo di Crank-Nicolson, scegliendo $\lambda = -42$, $t_{max} = 1$, $y_0 = 2$ e $h = 0.02$.

4. Scrivere la function `Heun.m` che implementa il metodo di Heun per la risoluzione del problema di Cauchy (4). La funzione `Heun.m` riceve in ingresso:

   - la funzione $f(t, y)$ definita come anonymous function (tramite il comando `@`). Si noti che f è una funzione di due variabili;
   - l'estremo t_{max} dell'intervallo di definizione;
   - il dato iniziale y_0;
   - il passo di avanzamento temporale h.

Si suppone che l'istante iniziale sia $t_0 = 0$. La funzione `Heun.m` restituisce il vettore `t_h` degli istanti temporali discreti e il vettore `u_h` dei valori della corrispondente soluzione approssimata. L'intestazione di `Heun.m` sarà quindi:

```
function [t_h,u_h]=Heun(f,t_max,y_0,h)
```

5. Utilizzando la function appena scritta, risolvere il problema (3) con il metodo di Heun, scegliendo $\lambda = -42$, $t_{max} = 1$, $y_0 = 2$ per il valore del passo $h = 0.02$.

6. Definiamo $e_h = \max_{n\in[0,1,\dots,N_h]} |y(t_n) - u_n|$, il massimo del modulo dell'errore compiuto approssimando la soluzione esatta $y(t_n)$ con la soluzione numerica u_n. Riportare, su un grafico in scala logaritmica, l'andamento di e_h, ottenuto utilizzando i metodi di Eulero all'indietro (funzione `eulero_indietro_pto_fisso`), Crank-Nicolson, Heun, il metodo di Runge-Kutta di ordine 4 (si usi la function `Runge_Kutta_4.m` fornita), al variare di h, utilizzando i passi temporali $h = [0.02, 0.01, 0.005, 0.0025, 0.00125]$. Confrontare gli ordini di convergenza ottenuti con quelli previsti dalla teoria.

3 Approssimazione di sistemi di equazioni differenziali ordinarie

Si considerino i seguenti due casi:

- *Sistema di equazioni differenziali ordinarie (EDO) del primo ordine*
  Si consideri il seguente sistema di m equazioni differenziali ordinarie nelle incognite $y_1 = y_1(t), \dots, y_m = y_m(t)$:

$$\begin{cases} y_1' = f_1(t, y_1, \dots, y_m) \\ \quad \vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad t \in (t_0, t_f), \\ y_m' = f_m(t, y_1, \dots, y_m) \\ y_1(t_0) = y_{0,1}, \ \dots, \ y_m(t_0) = y_{0,m}. \end{cases}$$

- *Equazione differenziale ordinaria di ordine p*
  Si consideri il caso di un'equazione differenziale di ordine p:

$$\begin{cases} y^{(p)}(t) = f(t, y(t), y'(t), \dots, y^{(p-1)}(t)) \qquad\qquad t \in (t_0, t_f), \\ y(t_0) = y_0, \ y'(t_0) = y_1, \ \dots \ , y^{(p-1)}(t_0) = y_{p-1}. \end{cases}$$

  Ponendo

$$w_1(t) = y(t), \ w_2(t) = y'(t), \ \dots \ , w_p(t) = y^{(p-1)}(t),$$

  l'equazione differenziale di ordine p può essere ricondotta al seguente sistema di p equazioni lineari di ordine 1:

$$\begin{cases} w_1' = w_2(t) \\ \quad \vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad t \in (t_0, t_f), \\ w_p' = f(t, w_1, \dots, w_p) \\ w_1(t_0) = y_0, \ w_2(t_0) = y_1, \ \dots \ , w_p(t_0) = y_{p-1}. \end{cases}$$

Per la risoluzione del sistema di equazioni differenziali ordinarie si possono applicare a ciascuna delle equazioni che compongono il sistema uno dei metodi di approssimazione visti per il caso scalare. In alternativa, Matlab® fornisce delle funzioni built-in per la risoluzione di sistemi di EDO del primo ordine. La sintassi da utilizzare per chiamare queste funzioni è

```
[t,y] = odeXX('fun',[t0 tf],y0)
```

dove: `y0` è il vettore delle condizioni iniziali; `[t0 tf]` è l'intervallo temporale in cui è definito il problema; `'fun'` è una funzione specificata dall'utente; `odeXX` è uno dei metodi disponibili in Matlab® (ad esempio, `ode23` o `ode45`; si veda la documentazione dei metodi per approfondimenti). L'output della funzione è composto dal vettore `t`, che contiene gli istanti temporali di discretizzazione selezionati dal metodo adattivo, e dalla matrice `y` che contiene i valori assunti dalla funzione approssimante negli istanti temporali `t`. L'utente deve quindi implementare la funzione `fun` che restituisce la valutazione dei termini a destra nel problema di Cauchy.

Esempio

Supponiamo di voler risolvere il seguente sistema di EDO del primo ordine avente dimensione $m = 2$:

$$\begin{cases} y_1'(t) = y_1(t) - (1 + \sin(\pi\, t))\; y_1(t)\, y_2(t) & t \in (0, 10), \\ y_2'(t) = y_2(t)\, y_1(t) - y_2(t) & t \in (0, 10), \\ y_1(0) = 2, \quad y_2(0) = 2. \end{cases}$$

Possiamo richiamare il programma built-in `ode45` con la seguente sintassi

```
[t,u] = ode45('fun_problema', [0 10], [2 2]);
```

avendo salvato, nel file `fun_problema.m`, la seguente funzione:

```
function fn = fun_problema( t, y )
[ n, m ] = size( y );
fn = zeros( n, m );
fn( 1 ) = y( 1 )  - ( 1 + sin( pi * t ) ) * y( 1 ) * y( 2 );
fn( 2 ) = y( 2 ) * y( 1 ) - y( 2 );
return
```

L'output della funzione `ode45` è composto dal vettore `t`, che contiene gli istanti temporali di discretizzazione, e dalla matrice `u` di due colonne, che contiene i valori *approssimati* dalle variabili y_1 e y_2 negli istanti considerati.

Esercizio 3.1

Si consideri il sistema massa-molla-smorzatore in Figura 3.
Come già sappiamo, la molla esercita una forza proporzionale allo spostamento dalla posizione di riposo $F_k = -kx$. Aggiungiamo al sistema una forza di attrito proporzionale alla velocità $F_c = -cx'$, e scriviamo il bilancio delle forze $F_k + F_c = mx''$. Si ottiene l'equazione differenziale di ordine 2 dell'oscillatore smorzato: $mx'' + cx' + kx = 0$. Dividendo per m e ponendo $\gamma = c/m$, $\omega^2 = k/m$ si ottiene:

$$x'' + \gamma x' + \omega^2 x = 0.$$

Si pongano $\gamma = 0.1$ e $\omega^2 = 1$.

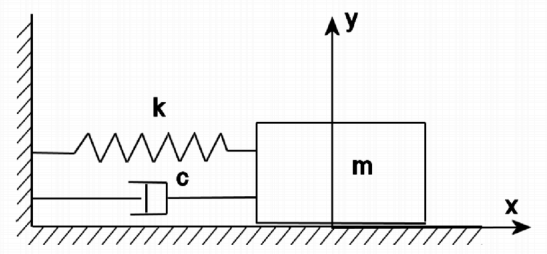

Figura 3: Sistema massa-molla-smorzatore

1. Dopo aver scritto l'equazione differenziale di ordine 2 sotto forma di un sistema di due equazioni differenziali di ordine 1, scrivere la `function mms` che restituisce la valutazione dei termini di destra del sistema.

2. Risolvere il problema nell'intervallo di tempo $(0, 100)$ con le condizioni iniziali $x(0) = 2$, $x'(0) = 0$ usando la `function ode45` built-in di Matlab® . Verificare graficamente che la soluzione del problema è costituita da un'oscillazione armonica di frequenza $\omega_1/2\pi$ con $\omega_1 = \sqrt{4\omega^2 - \gamma^2}/2$ smorzata nel tempo.

3. Aggiungere al sistema la forzante esterna $F_{\text{ext}}(t) = m\, A_0 \sin(\omega_f t)$, con $A_0 = 0.5$ e $\omega_f = 0.5$, ottenendo il seguente bilancio delle forze $F_k + F_c + F_{\text{ext}} = mx''$. Verificare graficamente che, a regime, la soluzione del problema è un'oscillazione armonica di frequenza $\omega_f/2\pi$.

Esercizio 3.2

Si consideri il problema di Keplero inerente due corpi puntiformi aventi masse m_1 e m_2 e posizioni $\mathbf{r}_1(t)$ e $\mathbf{r}_2(t)$ al tempo t in un sistema di riferimento inerziale e che interagiscono tramite la forza gravitazionale; assumiamo che il moto dei due corpi avvenga in un piano, ovvero $\mathbf{r}_i(t) = r_{i,x}(t)\,\hat{\mathbf{x}} + r_{i,y}(t)\,\hat{\mathbf{y}}$, per $i = 1, 2$. Vogliamo determinare l'orbita di Keplero del corpo 2 assimilabile al pianeta Terra, ovvero $\mathbf{r}_2(t)$, assumendo che $\mathbf{r}_1(t) = \mathbf{0}$ per ogni $t \geq t_0$, essendo il corpo 1 il Sole di massa $m_1 \gg m_2$. Ponendo dunque $m = m_1$, $\mathbf{r}(t) = \mathbf{r}_2(t)$, il modello matematico che descrive il fenomeno è rappresentato dal seguente sistema di EDO del secondo ordine in forma adimensionale:

$$\begin{cases} \mathbf{r}''(t) = -4\,\pi^2\, \dfrac{\mathbf{r}(t)}{\|\mathbf{r}(t)\|^3} & t \in (t_0, t_f), \\ \mathbf{r}(t_0) = \mathbf{r}_0, \\ \mathbf{r}'(t_0) = \mathbf{v}_0, \end{cases}$$

dove $\mathbf{r}_0$ è la posizione iniziale e $\mathbf{v}_0$ la velocità iniziale del corpo 2; $\mathbf{r}(t)$ esprime la posizione della Terra rispetto al Sole in unità astronomiche.

1. Si riscriva il sistema di EDO del secondo ordine come un sistema di EDO del primo ordine di dimensione m.

2. Posti $\mathbf{r}_0 = (1, 0)^T$ e $\mathbf{v}_0 = (0, -5.1)^T$, si risolva il problema nell'intervallo di tempo $(0, 5)$ usando opportunamente la `function ode45` di Matlab® . Si rappresentino l'andamento

della posizione $\mathbf{r}(t)$ approssimata vs. t e l'orbita del pianeta nel piano. Si usi la funzione come segue `[t, u] = ode45( ..., options );` avendo assegnato in precedenza `options = odeset('reltol',1e-6);`.

3. Si scriva una `function` Matlab® `eulero_avanti_sistemi.m` che implementi il metodo di Eulero in avanti per l'approssimazione di un sistema di EDO del primo ordine. L'intestazione della funzione sarà:

   ```
   [ t, u ] = eulero_avanti_sistemi( @twobody, [t0 tf], y0, Nh )
   ```

   dove `twobody` rappresenta il termine di destra del problema di Cauchy del primo ordine (da salvare nella funzione Matlab® `twobody.m`), `[t0 tf]` l'intervallo temporale, `y0` il dato iniziale (un vettore colonna di lunghezza pari ad m) e `Nh` il numero di sottointervalli in cui è suddiviso $[t_0, t_f]$; in uscita, la funzione ritorna il vettore `t` dei tempi discreti e la matrice `u` contenente le componenti della soluzione approssimata.

4. Si utilizzi la funzione Matlab® `eulero_avanti_sistemi.m` per risolvere il problema con i dati precedentemente indicati e per valori del passo $h = 10^{-5}$ (`Nh`$= 5{\cdot}10^5$); si rappresenti la soluzione numerica ottenuta. Si ripeta per $h = 10^{-3}$ e si commenti il risultato ottenuto.

Esercizio 3.3

Consideriamo un modello epidemiologico compartimentale di tipo SEIR per lo studio della dinamica di una malattia infettiva in un gruppo di individui (popolazione).
`https://it.wikipedia.org/wiki/Modelli_matematici_in_epidemiologia`
`http://gabgoh.github.io/COVID/index.html`
`https://www.epimox.polimi.it/`

Il modello SEIR considera in una popolazione di N individui i seguenti compartimenti: $S(t)$, il numero di individui suscettibili al tempo t; $I(t)$, il numero di individui infettivi; $E(t)$, il numero di individui esposti (per tener conto del periodo di incubazione per individui infetti, ma non ancora infettivi) e $R(t)$ il numero di recuperati o rimossi, ovvero individui che non rientrano più nella dinamica dell'epidemia (guariti, immuni, isolati, deceduti, etc...). Il modello SEIR corrisponde a un sistema di EDO del primo ordine nella forma seguente:

$$\begin{cases} \frac{dS}{dt} = -\beta\,\frac{S\,I}{N} & t \in (t_0, t_f), \\ \frac{dE}{dt} = +\beta\,\frac{S\,I}{N} - \alpha\,E & t \in (t_0, t_f), \\ \frac{dI}{dt} = +\alpha\,E - \gamma\,I & t \in (t_0, t_f), \\ \frac{dR}{dt} = +\gamma\,I & t \in (t_0, t_f), \\ S(t_0) = S_0,\ E(t_0) = E_0,\ I(t_0) = I_0,\ R(t_0) = R_0, \end{cases}$$

dove S_0, E_0, I_0 e R_0 sono i valori iniziali, β rappresenta il tasso di infezione o contagio, α il tasso di incubazione (α^{-1} rappresenta il tempo di incubazione medio), γ il tasso di rimozione (γ^{-1} rappresenta il tempo infettivo medio). Il valore del numero di riproduzione di base è $r_0 = \frac{\beta}{\gamma}$, il numero di infetti secondari che ogni infezione produce. Osserviamo che, essendo

$S(t) + E(t) + I(t) + R(t) = N$ per ogni $t \in [t_0, t_f]$, il precedente sistema di 4 equazioni può essere ridotto a un sistema di EDO con $m = 3$ equazioni in S, E ed I, da cui poi ricavare R.

1. Si riscriva il modello SEIR come un sistema di $m = 3$ EDO del primo ordine. Posti $N = 10^7$, $E_0 = 100$, $I_0 = 30$, $R_0 = 0$, $\beta = 0.7586$ giorni^{-1}, $\alpha^{-1} = 5.2$ giorni, $\gamma^{-1} = 2.9$ giorni, $t_0 = 0$ e $t_f = 300$ giorni, si risolva il problema tramite il metodo di Eulero in avanti con passo $h = 0.25$ giorni usando la funzione `eulero_avanti_sistemi.m`. Che percentuale della popolazione ($100\, R_\infty/N$) rientra nel gruppo dei rimossi al tempo $t = t_f$?

2. Si risolva ora il problema assumendo che il valore β passi da 0.7586 a 0.25 giorni^{-1} a partire dal giorno $t = 80$ a seguito dell'attuazione di una politica di isolamento della popolazione. Che percentuale della popolazione rientra nel gruppo dei rimossi al tempo $t = t_f$? Cosa succede se il valore di β risale a 0.65 giorni^{-1} al giorno $t = 120$?

3. Usando i dati di cui al punto 1, si studi l'assoluta stabilità del metodo di Eulero in avanti, sapendo che $\lim_{t\to+\infty} S(t) = S_\infty = N - R_\infty$ e $\lim_{t\to+\infty} E(t) = \lim_{t\to+\infty} I(t) = 0$.