Progetto di Calcolo Numerico 2

Alyssa Pezzutti

20 luglio 2020

Abstract

In questa relazione studieremo il problema 3 tratto dall'elenco di temi d'esame fornito dal professore, motivando le scelte che abbiamo fatto riguardo ai metodi e alla stima dell'errore usati.

1 Pendolo accoppiato.

Un pendolo accoppiato può essere modellato dalle equazioni differenziali

$$\varphi_1'' = -\frac{\sin\varphi_1}{l_1} - \frac{c_0 r^2}{m_1 l_1^2}(\sin\varphi_1 - \sin\varphi_2)\cos\varphi_1 + f(t)$$

$$\varphi_2'' = -\frac{\sin\varphi_2}{l_2} - \frac{c_0 r^2}{m_2 l_2^2}(\sin\varphi_2 - \sin\varphi_1)\cos\varphi_2$$

Come richiesto approssimeremo la soluzione esatta sull'intervallo [0, 250] per

$$l_1 = l_2 = 1, m_1 = 1, m_2 = 0.99, r = 0.5, c_0 = 0.5$$

$$\varphi_1(0) = \varphi_1'(0) = \varphi_2(0) = \varphi_2'(0) = 0$$

$$f(t) = \begin{cases} \sqrt{1-(1-t)^2}, & \text{se } |t-1| \leq 1, \\ 0 & \text{altrimenti.} \end{cases}$$

Si tratta di un sistema non lineare di ODE del II ordine della forma

$$\underline{\varphi}'' = g(t, \underline{\varphi}), \text{ con } \underline{\varphi} = (\varphi_1, \varphi_2)$$

$g : \mathbb{R} \times \mathbb{R}^2 \to \mathbb{R}^2$. g è solamente continua in [0, 250] in quanto la funzione $f(t)$ non ammette derivata prima in $t = 2$ (in $t = 0$ invece le condizioni iniziali garantiscono che la derivata esista e valga 0). La soluzione esatta $\underline{\varphi}$ sarà dunque una funzione di classe $C^2(\mathbb{R}^2)$.
Innanzitutto, trasformiamo il sistema in un sistema del I ordine. Introduciamo quindi le variabili

$$y_1 = \varphi_1' \quad \text{e} \quad y_2 = \varphi_2'$$

e riscriviamo il sistema in questo modo:

$$\varphi_1' = y_1$$
$$y_1' = -\frac{\sin\varphi_1}{l_1} - \frac{c_0 r^2}{m_1 l_1^2}(\sin\varphi_1 - \sin\varphi_2)\cos\varphi_1 + f(t)$$
$$\varphi_2' = y_2$$
$$y_2' = -\frac{\sin\varphi_2}{l_2} - \frac{c_0 r^2}{m_2 l_2^2}(\sin\varphi_2 - \sin\varphi_1)\cos\varphi_2$$

2 Metodi

Ci siamo orientate fin da subito su dei metodi espliciti, infatti, poiché sicuramente le componenti φ_1 e φ_2 avrebbero presentato numerose oscillazioni essendo dei pendoli, abbiamo pensato che dei metodi a passo fisso potessero approssimare adeguatamente il problema, con dei costi relativamente bassi rispetto per esempio a dei metodi impliciti.
Abbiamo comunque calcolato lo Jacobiano di g (che per brevità non riporto qui ma si trova nel file `problemi3.cpp` allegato) e gli autovalori nell'origine, che riportiamo sotto, per poter provare dei metodi impliciti e indagare il dominio di stabilità del problema.

$$\lambda_1 = -1, \quad \lambda_2 = 1.119, \quad \lambda_3 = 0.0030 + 1.0575i \quad \lambda_4 = 0.0030 - 1.0575i$$

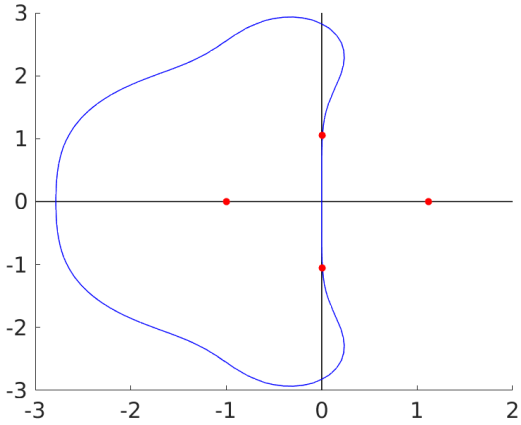

Figure 1: Autovalori (in rosso) e regione di assoluta stabilità di RK4

Vista la posizione iniziale degli autovalori nel piano di Argand-Gauss, abbiamo pensato che sia dei metodi impliciti, che dei metodi di Runge-Kutta di ordine abbastanza alto avrebbero reso i risultati migliori.
Provando dei metodi impliciti però, era chiaro fin da subito che nonostante una buona approssimazione della soluzione esatta, il loro costo in termini di

valutazioni di g fosse troppo alto, perciò li abbiamo esclusi.
Per sfizio, abbiamo anche provato dei metodi immersi, ma con l'aspettativa che la ricerca di τ ottimale ad ogni passo li rendesse estremamente costosi. Questa previsione si e' rivelata corretta per Heun e Fehlberg adattivi. Dormand-Prince (DP87) invece sembrava una buona scelta, ma per tempi $t \in [0, 11]$ il grafico non presentava oscillazioni come invece avrebbe dovuto.
Alla fine la scelta è ricaduta su Runge-Kutta 4, che solitamente fornisce un buon compromesso tra accuratezza e costo. Per soli 1600 passi infatti (6400 valutazioni di g) siamo riuscite a ottenere un'approssimazione qualitativamente corretta, i.e. in cui entrambe le componenti presentano dei battimenti.

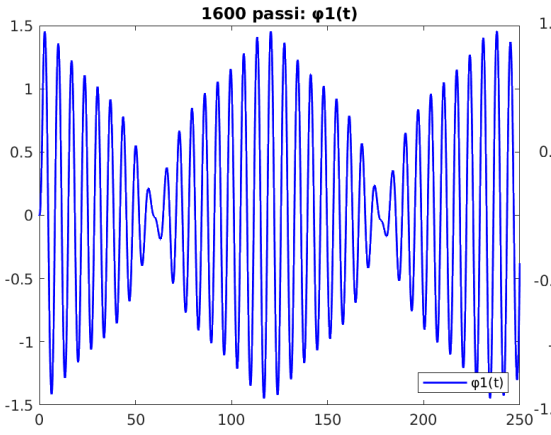

Figure 2:
φ_1, RK4 con 1600 passi

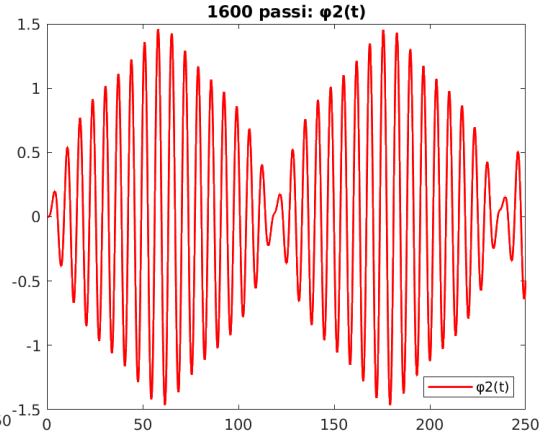

Figure 3:
φ_2, RK4 con 1600 passi

3 Stima dell'errore

Naturalmente non ci si può accontentare di un approccio qualitativo, dunque abbiamo cercato un modo per stimare l'errore. Non essendo in possesso della soluzione esatta, abbiamo deciso di sfruttare la stima introdotta nel Laboratorio 4

$$|u(T) - U_N| \approx \frac{\tilde{N}^p}{|\tilde{N}^p - N^p|} |U_{\tilde{N}} - U_N|$$

dove u e U sono rispettivamente la soluzione esatta e approssimata, N e $\tilde{N}$ sono il numero di passi usati, e p è l'ordine del metodo. Inoltre abbiamo scelto di usare $\tilde{N} = 2\mathrm{N}$, in modo che la stima non dipendesse dal numero di passi. Abbiamo quindi ottenuto

$$|u(T) - U_N| \approx \frac{2^p}{|2^p - 1|} |U_{2N} - U_N|$$

Questa stima però si riferisce solo all'ultimo punto della soluzione approssimata ($t = 250$), dunque non è adatta ad una funzione periodica, e non rappresenta accuratamente gli errori visibili nel grafico per $t \in [0, 250)$.
Abbiamo quindi usato come stima dell'errore globale la quantità

$$\frac{2^p}{|2^p - 1|} \max_{1 \le k \le N} \{||U_{2k} - U_k||_2\}$$

e come stima dell'errore globale su ciascuna componente

$$\frac{2^p}{|2^p - 1|} \max_{1 \le k \le N} \{|\varphi_{i,2\,k} - \varphi_{i,k}\,|\}$$

$i = 1, 2$, dove $U = (\varphi_1, \varphi_2, \varphi_1', \varphi_2')$, e p è l'ordine teorico del metodo.
L'algoritmo in Matlab è il seguente:

```
n = 400;   %n iniziale
metodo = 'rk4_';
p = 4;
X = load(strcat(metodo,string(n)));
C = 2; %costante moltiplicativa dei passi

i = 1; %indice di riga della matrice err
while (n <= 51200)
Y = load(strcat(metodo,string(C*n)));
err(i,1) = n;

for (k = 1:n+1)
diff(1:4) = abs((X(k,2:5)-Y(C*(k-1)+1,2:5))); %C*(k-1)+1 e' l'indice
    di riga nel file '2*n' corrispondente all'indice k nel file 'n'
N(k) = norm(diff);
end

err(i,2) = (C^p/(C^p-1))*max(diff(1));
err(i,3) = (C^p/(C^p-1))*max(diff(2));
err(i,4) = (C^p/(C^p-1))*max(norm(diff));
i = i+1;
n = C*n;
X = Y;
end
```

Abbiamo scritto anche una versione più efficiente dell'algoritmo che si trova nel file `errore2_efficiente.m` La versione riportata però è più "leggibile" e perciò è stata preferita. Per $N \approx 100000$ l'esperienza conferma che questa versione funziona ancora bene.

N	Errore 1	Errore 2	Errore globale
400	1.3865e-01	9.1693e-01	2.4989e+00
800	7.5904e-01	5.0382e-01	1.2908e+00
1600	7.9944e-03	8.2925e-03	4.8492e-02
3200	3.3842e-03	1.1099e-03	1.2526e-02
6400	3.0855e-03	9.5499e-04	1.1819e-02
12800	4.8015e-03	2.2551e-03	2.1548e-02
25600	1.8940e-04	7.0261e-05	7.2882e-04
51200	5.8419e-05	1.9824e-05	2.1713e-04

Table 1: Tabella degli errori sulle componenti 1 e 2

Riportiamo adesso i risultati e i costi del metodo esplicito a passo fisso Runge-Kutta 4:
dove per ciascun N fissato questo metodo costa 4N valutazioni di g.

Purtroppo il metodo non riesce a raggiungere l'ordine che dovrebbe avere ($p = 4$), fenomeno che pensiamo sia dovuto soprattutto alla poca regolarità della soluzione esatta $\underline{\varphi}$.

4 Conclusioni

Pensiamo che il metodo Runge-Kutta 4 sia ottimo per un'approssimazione qualitativamente corretta seppur poco precisa della soluzione esatta, in quanto per soli 400 passi i grafici delle componenti raffigurano già dei pendoli. Rispetto ad altri metodi che hanno reso gli stessi risultati, questo è stato senza dubbio il più efficiente.
I grafici per i vari passi provati possono essere trovati nella cartella "immagini" allegata al progetto.
Alleghiamo inoltre i file matlab `calcoloerrore_grafici.m, errore2_efficiente.m`