Laboratorio 12

Obiettivi

Approssimare il problema di Cauchy per un SISTEMA di E.D.O.:

$f: \Re^{d+1} \longrightarrow \Re^d$

$u'(t) = f(t,u(t)), \quad t_0 \le t \le T, \quad u(t) \in \Re^d$

$u(t_0) = v, \quad v \in \Re^d$

con un metodo **Multi – Passo esplicito**

Metodi multi passo

Assegnati i due polinomi caratteristici di grado k

$$\rho = \sum_{i=0}^{k} \rho_i x^{k-i} \quad , \quad \sigma = \sum_{i=0}^{k} \sigma_i x^{k-i} \quad ,$$

un metodo **Multi – Passo** (per ora esplicito) **a k passi con τ fisso** si scrive:

$$\sum_{i=0}^{k} \rho_i U_{n-i} = \tau \sum_{i=0}^{k} \sigma_i f_{n-i}$$

dove si è posto $f_i = f(t_i , U_i)$.

Nota 1: allo scopo di utilizzare notazioni i cui indici coincidano con quelli che si useranno in C++, qui **le notazioni non coincidono con quelle utilizzate dal docente**: qui ρ_0 e σ_0 sono i coefficienti direttivi e non i termini noti. I coefficienti dei polinomi saranno memorizzati in due vettori, ro e sigma. (vedi il file multi_step_coefficienti.cpp allegato, contenente i coefficienti di alcuni metodi di Adams)

Nota 2: k è il numero dei passi all'indietro e il grado dei polinomi. La richiesta che il metodo sia esplicito si traduce nella condizione $\sigma_0 = 0$, ossia sigma[0]=0, ciò farà la differenza con il caso implicito. Per semplicità di programmazione e uniformità con il caso implicito, considereremo σ un polinomio di grado k con coefficiente direttivo nullo.

Esempio

Al metodo a due passi di Adams-Bashforth

$$u_n = u_{n-1} + \frac{\tau}{2}[3f_{n-1} - f_{n-2}]$$

Si associano i vettori

ro = [1, -1, 0]

sigma = [0, 3/2, -1/2]

Alcune indicazioni su come impostare il lavoro:

- Può essere conveniente normalizzare i coefficienti di ρ e σ in modo che $\tilde{\rho}_0 = 1$ e, riscrivere il metodo come:

$$U_n = -\sum_{i=1}^{k} \tilde{\rho}_i U_{n-i} + \tau \sum_{i=0}^{k} \tilde{\sigma}_i f_{n-i}$$

- Per metodi con un numero di passi k > 1 sarà necessario calcolare $(k-1)$ valori iniziali aggiuntivi con un metodo diverso: si consiglia un Runge-Kutta di ordine elevato.

- Ad ogni passo si calcola un solo valore nuovo delle f_i, precisamente f_{n-1}, gli altri si salvano dai passi precedenti. Sarà utile avere 2 matrici Old_U e Old_effe che memorizzano i vecchi valori U_i e f_i che servono nei passi successivi.

- È utile avere una funzione che copia un vettore in un altro.

Schema di lavoro per l'implementazione del metodo

NB: in **grassetto rosso** si indica ciò che si intende fare nelle righe sottostanti

N.B. il parametro **k**, che nelle formule indica il grado dei polinomi ρ e σ e il numero di passi all'indietro utilizzati, nello schema sottostante è tradotto con la variabile **passi**

// Dimensionamenti:

- tutto ciò che serve per un Runge-Kutta esplicito
- ro[passi+1], sigma[passi+1]

- Old_U[passi][d], Old_effe[passi][d],
- …

// Inizializzazioni:

- $t = t_0$, $u = v$, …
- ro e sigma (vedi apposita funzione)
- normalizzazione di ro e sigma
- STAMPA t, u

// Calcolo dei valori iniziali aggiuntivi con un metodo RK esplicito:

n = 0, …< (passi-1) (ciclo ridotto sul tempo)

- memorizzo u in n-esima riga di Old_u
- calcolo f(t,u) memorizzandola in n-esima riga di Old_effe
- eseguo un passo di RK (NB: K[0] coincide con f(t,u) appena calcolata)
- incremento il tempo
- STAMPA t, u

end ciclo ridotto

// Vero ciclo sul tempo:

n = passi, …., N

- copio u in Old_u alla riga di indice **(n-1)%passi**
- calcolo f(t,u) memorizzandola in Old_effe alla riga di indice **(n-1)%passi**
- calcolo u nuovo usando i valori in Old_u e Old_effe associando i coefficienti nell'ordine giusto: dimensiono

su[d] , sf[d] vettori ausiliari per le due somme

coefu[passi], coeff[passi] vettori ausiliari dei coefficienti

for int j =1, … passi

int **k = (n-j)%passi**;

coeff [k] = sigma [j];

coefu [k] = ro[j];

end j

prodotto vettore riga * matrice: coefu*Old_u → su

prodotto vettore riga * matrice: coeff*Old_f → sf

u = - su + tau*sf; // qui sottinteso un ciclo su componenti vettore

- incremento il tempo
- STAMPA t, u

end ciclo su n

Esercizio 12.1

Testare il buon funzionamento del programma sul problema di Van der Pol.

Provare con AB3 e AB4 per N = 100, 1000, 10000. Verificare che l'ordine sperimentale coincida con quello teorico.

Esercizio 12.2

Inizializzare un metodo multi_passo di ordine p > 2 con un metodo RK di ordine q < (p – 1) applicati ad un problema sufficientemente regolare. Analizzare le conseguenze.

Esercizio 12.3

Approssimare il problema di Dahlquist con un metodo di Adams-Bashforth.. Provare con gli stessi valori di λ e N suggeriti nell'esercizio 9.2. Legare i risultati ottenuti ai domini di stabilità dei metodi (vedi funzione di Matlab *msregstab.m* scaricabile dal sito del corso).

Esercizio 12.4

Approssimare il problema di Dahlquist con λ = i, T = 10, con il metodo Leap Frog:

$$u_n = u_{n-2} + 2\tau\, f_{n-1}$$

N = 10, 100, 1000, 10000. Visualizzare la traiettoria della soluzione approssimata e calcolare l'errore finale.

Ripetere le medesime prove per λ = -1. Fissato N = 10000, prolungare il tempo T a 20 (visionare il grafico!!!).

Mettere in relazione i risultati con il dominio di stabilità del metodo.