Nona Esercitazione: Equazioni differenziali e differenze finite

Varie di programmazione

Esercizio 1 *Equazione della trave mediante le differenze centrate*

Costruire un m-file funzione di **c**, **p**, **t** che calcoli e disegni la soluzione approssimata del problema differenziale

$$\begin{cases} y'' + c(t)y(t) = p(t) \\ y(t_a) = 0 \qquad y(t_b) = 0 \end{cases}$$

usando le differenze finite nell'intervallo $[t_a, t_b]$ con passo h. Con **t** si intenderà l'array $[t_a\ t_a + h\ ...\ t_b]$ cioè l'intervallo diviso con passo h. Con **c** e **p** si intenderanno le funzioni $c(t)$ e $p(t)$ già tabulate su **t**.
Dobbiamo trovare **y**.

Occorrerà risolvere il sistema lineare tridiagonale

$$\begin{cases} \dfrac{y_3 - 2y_2 + y_1}{h^2} + c_2 y_2 & = & p_2 \\ \dfrac{y_4 - 2y_3 + y_2}{h^2} + c_3 y_3 & = & p_3 \\ \dots & \dots & \dots \\ \dfrac{y_n - 2y_{n-1} + y_{n-2}}{h^2} + c_{n-1} y_{n-1} & = & p_{n-1} \end{cases}$$

nelle incognite $y_2\ ,\ ...\ ,\ y_{n-1}$.

La matrice è tridiagonale. La diagonale è $[c_2h^2 - 2\ ,\ ...\ ,\ c_{n-1}h^2 - 2]$
Il termine noto è $[p_2h^2\ ,\ ...\ ,\ p_{n-1}h^2] - [y_1, 0, 0, ..., 0, y_n]$ (trasposto)

Usare il comando generalizzato **diag** (vedi **help diag**) per costruire la matrice.

Provare inizialmente usando come **c** la matrice di -1 e come **p** la matrice [1 1 ...1 2 2 2] (circa a metà).
t_a e t_b qualunque (per es. 0 e 1)

Disegnare la funzione con vari **p** e confrontare i grafici ottenuti.

Esercizio 2 *Equazione del calore con le differenze in avanti*

L'equazione differenziale del calore nella sua forma più semplice è

$$\begin{cases} \dfrac{\partial u}{\partial x} - c\dfrac{\partial^2 u}{\partial t^2} \\ u(x,0) = u0(x) \text{ funzione definita in } [a,b] \\ u(a,t) \equiv u(b,t) \equiv 0 \end{cases}$$

Occorrerà innazitutto che sia definita, come funzione esterna, la funzione $u0$.

Costruire un m-file funzione di **x**, **t**, **c** che dia come risultato una matrice **u** (ovvero la funzione $u(x,t)$ nel rettangolo $[a,b] \times [0,T]$).

c costante positiva.
x intervallo $[a,b]$ già diviso secondo un passo dx.
t intervallo $[0,T]$ già diviso secondo un passo dt (si suggerisce $dt = dx^2/2$).

Si calcola il formato della matrice **u** e sarà **M**×**N**.

Si inizializza la matrice **u** con tutti zeri.

Si inizializza la prima riga della matrice **u** con $u0$.

Si definisce la costante **alpha** $\left(\alpha = c\,\dfrac{dt}{dx^2}\right)$

Quindi si usa la formula $\qquad u_i^{n+1} = u_i^n + \alpha(u_{i-1}^n - 2u_i^n + u_{i+1}^n)$

Sarà necessario un loop **for...end** per la **n**, ma non per la **i**. Per calcolare **u** è bene, per maggior accuratezza, che dt sia piccolo come suggerito. Per il grafico della funzione sarà bene ridurre un po' la **u** che è troppo grande con un comando del tipo **U=u(1:dtt:end,1:dxx:end);**

Per collaudare l'algoritmo provare a usare
- La funzione $u0(x) = x(1-x)$ nell'intervallo $[0,1]$
- Poi la funzione $u0(x) = x(1-x)(100x^2 - 100x + 25)$ nell'intervallo $[0,1]$

Esercizio 3 *Il commesso viaggiatore*

Il problema è il seguente: Sono dati sette punti nel piano, per esempio quelli a lato che possono essere pensati come città da visitare.
Si tratta di trovare l'itinerario più breve che partendo da $(0, 0)$ passi per tutti i sette punti.
Iniziare col costruire la matrice **d** 8×8 delle distanze tra i vari punti. Il programma per trovare l'itinerario (nel peggior modo, cioè provando tutti i 5040 percorsi possibili) può essere articolato così:

x	y
3	0
2	1
0	2
5	2
4	4
1	5
6	7

```
% ---- COSTRUIRE d -----
towns=7 % numero città da visitare
w=perms(1:towns); % costruisce tutte le possibili permutazioni dei
         % numeri da 1 a towns
         % Per towns=7 è una matrice di 5040 righe.  Non scrivetela !!!
minimo=10000 % Inizializzare minimo con un numero alto
for id=1:length(w)
     dist=d(1,w(id,1)) % inizializza la distanza (da (0,0) alla prima città)
     for jd=1:towns-1 % i segmenti sono towns-1
              dist=dist+d(..  , ..)  % inserire gli indici giusti...
     end
     if....% vedo se la distanza è minima.  Se l'ho trovata, devo conservare
           % la permutazione per la fine e aggiornare il valore di minimo
     end
end
plot(....)
```

Esercizio 4 *Calendario: calcolare il giorno della settimana di una data.*

Premessa: La funzione modulo (**mod**). Se a, b sono numeri interi positivi, con $a \text{ MOD } b$ si indica il resto della divisione di a per b.
Per esempio $10 \text{ MOD } 7 = 3 \quad 35 \text{ MOD } 24 = 11 \quad 5 \text{ MOD } 8 = 5$
In MatLab la funzione è **mod(a,b)**

Creare un funzione **calendario(g,m,a)** che calcola il giorno della settimana di g/m/a.

Si comincia col calcolare il giorno in cui cade il primo gennaio nell'anno **a**
Si parte da un anno in cui il primo gennaio cadeva di domenica p.es. 1928.
Si calcola quanti giorni in più ci sono in questo modo: **d = a-1928**

Si divide **d** per 4 e si ottiene il quoziente **q** (che calcola quanti anni bisestili ci sono tra i due anni, compreso 1928)
Se la divisione è esatta (comando **frac**) si aggiunge a **d** il quoziente della divisione per 4: **d=d+q** (l'anno in questione non va aggiunto)
Altrimenti si aggiunge il quoziente +1 : **d=d+q+1**.

Il primo gennaio dell'anno **a** cadrà nel giorno **d** (0=domenica, 1=lunedì etc.)
Si riduce **d** modulo 7 per il caso in cui **d** superi 7.

Per calcolare il primo giorno del mese cercato si usano le seguenti tabelle: (usare il comando **switch**)

Anni normali	**Anni bisestili**
0 = Gen Ott	0 = Gen Apr Lug
1 = Mag	1 = Ott
2 = Ago	2 = Mag
3 = Feb Mar Nov	3 = Feb Ago
4 = Giu	4 = Mar Nov
5 = Set Dic	5 = Giu
6 = Apr Lug	6 = Set Dic

cioè aggiungere 1 a mesi di anni normali tranne Gen Feb

Ovvero si aggiunge il numero dato al giorno del primo gennaio (e si riduce modulo 7).

A questo punto dovrebbe essere facile calcolare qualunque giorno di qualunque anno dal 1/1/1928 al 31/12/2099 (per altri periodi sono necessarie modifiche che tengano conto del fatto che 1900 e 2100 non sono anni bisestili anche se divisibili per 4).