Metodi Numerici per il Calcolo

Esercitazione 3: Numeri Finiti e libreria anmglib_4.1 per il disegno

A.A.2024/25

Scaricare dalla pagina web del corso l'archivio matlab_mnc2425_3.zip e scompattarlo nella propria home directory. Verrà creata una cartella con lo stesso nome contenente script e function utili per questa esercitazione che ha come obiettivo sperimentare l'aritmetica finita (Numeri Finiti) e introdurre la libreria `anmglib_4.1` per il disegno in Matlab.

A. Numeri Finiti in precisione BASIC single e BASIC double

Matlab usa di default la precisione double e si può passare in precisione single mediante l'utilizzo della funzione `single()`. Poter alternare in un codice le precisioni single e double può essere molto utile a fini didattici.
I seguenti esercizi vogliono mettere in pratica alcuni concetti visti a lezione per rafforzare la propria comprensione; i vari script vanno modificati e rieseguiti più volte.

1. Lo script `scompute_u.m` calcola l'unità di arrotondamento U, sia in precisione single che double, mediante la seguente definizione operativa (ANSI/IEEE std.754):

   U è il più grande numero finito positivo tale che $fl(U + 1) = 1$**.**

   Analizzare l'implementazione della definzione operativa e verificare i risultati prodotti nei casi single e double.

2. Ciclo while e numeri gradual underflow. Analizzare lo script `sfiniti.m` che implementa un piccolo ciclo sia in versione BASIC single che BASIC double (vedi commenti) e stampa ad ogni iterazione un numero finito.

   (a) Prima di eseguire lo script prevedere cosa verrà prodotto in stampa.

   (b) Eseguire lo script, analizzare i risultati e individuare i numeri finiti **gradual underflow** dello standard ANSI/IEEE-754.

   (c) Modificare la condizione del ciclo `while` da `x>0` all'equivalente `x+1>1`; l'output rimane lo stesso'? Spiegare cosa succede.

3. Nello script `sexpression.m` viene effettuato il calcolo della semplice espressione

$$y = ((1 + x) - 1)/x$$

che dovrebbe sempre produrre come risultato il valore esatto 1, invece a seconda del valore assegnato ad x si ottengono risultati inattesi. A lezione si è applicata l'analisi in avanti degli errori per avere una stima dell'errore algoritmico di questa espressione e si è ottenuto che:

$$E_{Alg} \approx \frac{1}{x}|\epsilon_1| + 3U, \quad |\epsilon_1| < U.$$

Sperimentare differenti valori per x (reali o finiti) e dedurre per quali il risultato sarà corretto e per quali no e se i risultati sono coerenti con quanto stimato come E_{Alg}. (Sugg. utilizzare lo script `sconv_dec2bin.m` che permette di convertire un numero decimale in base 2 e controlla se è rappresentabile in modo esatto).

4. Approssimazione della derivata. Sia f una funzione continua e derivabile e cerchiamo di approssimare il valore della derivata mediante il limite del rapporto incrementale

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

L'idea è di valutare il rapporto incrementale per un certo valore di h, ma quanto piccolo? Ponendo $h = 0$ si ottiene $0/0$, cioè NaN. Lo script `fidiff.m` implementa il calcolo del rapporto incrementale e prova valori di h da 10^0 a 10^{-14} assumendo come x il valore 1.0, se non ne viene fornito uno differente. Lo script fa uso di precisione double. Si vede che l'approssimazione migliora al diminuire di h, ma quando h diventa troppo piccolo, l'approssimazione comincia a peggiorare. Che tipo di errore viene commesso?

B. Libreria `anmglib_4.1` per il Disegno

Si esegua lo script `add_path` presente nella cartella per poter utilizzare le funzioni della libreria `anmglib_4.1`. Ogni esercizio consiste nel realizzare uno script che richiami opportunamente alcune fra le seguenti funzioni della libreria per realizzare il disegno richiesto:

```
open_figure            axis_plot              point_plot
point_trans            point_trans_plot       line_plot
get_mat2_rot.m         get_mat2_trasl.m       get_mat2_scale.m
circle2_plot           circle2_trans_plot     rectangle_plot
rectangle_trans_plot
```

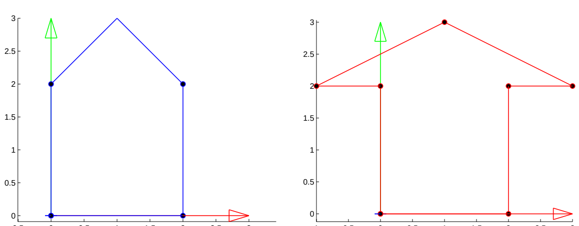
1. Si legga il file `paperino.txt` visto la scorsa esercitazione e si disegni la spezzata insieme al suo bounding-box e al sistema di assi acrtesiani utilizzando le function `point_plot` e `rectangle_plot` della libreria. Lo script si chiami `sload_plot.m`.
   Successivamente si riempia la spezzata con un colore usando la function `point_fill` della libreria.

2. Definire una lista di punti 2D per rappresentare un quadrato con un vertice nell'origine e lato 2 e disegnarlo insieme agli assi del sistema cartesiano. Definire poi la matrice di rotazione di un angolo α intorno all'origine e applicarla al quadrato. Lo script si chiami `ssquare_trans.m`; eseguirlo per $\alpha = \pi/3, 2\pi/3, 3\pi/3, 4\pi/3, 5\pi/3$, poi modificare il codice per disegnare insieme tutti i quadrati ruotati.

3. Definire una matrice composta per ruotare il quadrato dell'esercizio precedente rispetto al suo baricentro. Lo script si chiami `ssquare_rot.m`. (Attenzione: si realizzi una function per calcolare il baricentro di un oggetto generico definito da n punti/vertici distinti; non si consideri l'ultimo punto se coincide con il primo). Si ripeta lo stesso esercizio per scalare il quadrato rispetto al suo baricentro.

4. Si realizzi uno script per disegnare una circonferenza di centro l'origine e raggio 5; disegnare poi 12 circonferenze di raggio 1.4 aventi come centri punti equispaziati sulla circonferenza precedente. Lo script si chiami `scircles_plot.m`.

5. Definire e disegnare gli oggetti 2D mostrati in Figura 1(a) e 1(b). Gli script si chiamino `sfig2_trans2D.m` e `sfig3_trans2D.m`.



(a) (b)

Figura 1: "casa 2D" di 5 vertici e "freccia 2D" di 7 vertici.