UNIVERSITÀ DEGLI STUDI DI PADOVA

Laboratorio di Calcolo Numerico
Laboratorio 3: Algoritmi stabili e instabili, Bisezione

Claudia Zoccarato

E-mail: claudia.zoccarato@unipd.it

Dispense: Moodle Dipartimento ICEA

22 Marzo 2017

Vettori in MATLAB

- Finora abbiamo pensato alle variabili utilizzate come semplici valori numerici (variabili scalari). In realtà, in MATLAB ogni variabile è una struttura di tipo vettoriale (ARRAY).
- ARRAY $->$ lista di valori ordinati secondo uno o più indici.
- ARRAY ad un indice $->$ VETTORE
- ARRAY a due indici $->$ MATRICE
- Esempio (array ad un indice):

#1	#2	#3	#4	#5
1	-1	3	4	5

Vettori in MATLAB

- Memorizzare un vettore **riga** in MATLAB:
  `>> x = [1 -1 3 4 0]`
  Gli spazi delimitano le singole componenti del vettore riga
- Memorizzare un vettore **colonna** in MATLAB:
  `>> x = [1; -1; 3; 4; 0]`
  I punti e virgola delimitano le singole componenti del vettore colonna
- Esempio: visualizzare il valore della quarta componente del vettore (riga o colonna):
  ```
  >> x(4)
  ans =
         4
  ```
- Esempio: assegnare un valore alla terza componente del vettore (riga o colonna):
  `>> x(3) = -2`
- Utilizzare i comandi `who` e `whos` per verificare quali sono le variabili presenti nell'ambiente di lavoro e le informazioni sulla loro dimensione, spazio occupato in memoria e tipo di variabile.

Vettori in MATLAB - Notazione due punti

- Costruzione di vettori equispaziati:
  vettore = inizio:passo:fine

  ```
  >> x = 0:0.1:0.5
  x =
      0 0.1000 0.2000 0.3000 0.4000 0.5000
  ```

- Comando `linspace`:
  linspace(inizio,fine,numero di punti)

  ```
  >> x = linspace(0,0.5,6)
  ```

  Il numero di punti è opzionale, se omesso viene posto uguale a 100.

- Estrarre parte delle compenenti di un vettore:

  ```
  >> y = x(2:4)
  y =
      0.1000 0.2000 0.3000
  ```

Grafico di funzioni: comando plot

Visualizzare il grafico di una funzione NON predefinita in MATLAB

1. Vettorizzazione della funzione
   ```
   >> x = linspace(0,pi,10);
   >> y =
   (15120-6900*x.^2+313*x.^4)./(15120+660*x.^2+13*x.^4);
   ```
2. Operazioni vettoriali: i valori della funzione in corrispondenza della successione di punti definita dal vettore x sono creati con una singola istruzione vettoriale e assegnati al vettore y.
3. Comando `plot` (principale function grafica di MATLAB)
   ```
   >> plot(x,y)
   ```
   Produce il grafico dei punti (x_i, y_i).
   I due vettori devono avere la stessa lunghezza.

Definizione di funzioni: funzioni 'anonime'

1. Se vogliamo definire in MATLAB una funzione del tipo:
   $f(arg1, arg2, ..) = espressione$
2. La sintassi è la seguente:
   `>> f = @(x) (15120-6900*x.^2+313*x.^4)./(15120+660*x.^2+13*x.^4)`
3. Dopo il carattere `@` è indicata la variabile in ingresso della funzione (tra parentesi)
4. Posso valutare la funzione in $z = 0$:
   ```
   >> f(z)
   ans =
         1
   ```

Precisione numerica

Quando si opera in aritmetica finita, non sempre un'operazione tra due o più numeri macchina produce un risultato. In particolare si ha:

- **overflow** quando l'operazione produce come risultato un numero più grande del massimo numero rappresentabile
- **underflow** quando l'operazione produce come risultato un numero più piccolo del minimo numero rappresentabile

`realmax`	massimo numero macchina positivo
`realmin`	minimo numero macchina positivo
`eps`	precisione di macchina $(^*)$
`Inf`	∞, ovvero numero maggiore di `realmax`
`-Inf`	$-\infty$, ovvero numero minore di `-realmax`
`NaN`	Not-a-Number, tipicamente il risultato di operazioni illecite come $0 * \infty$, $0/0$ e ∞/∞

$(^*)$ Il più piccolo numero che sommato a 1 fornisce un numero maggiore di 1.

Stabilità numerica - schema instabile

Vogliamo scrivere un programma per il calcolo dei seguenti integrali I_n:

$$I_n = \frac{1}{e}\int_0^1 x^n e^x \, dx, \qquad n = 0, \dots, 20. \tag{1}$$

Dall'integrazione per parti si ottiene la seguente formula ricorrente:

$$I_n = 1 - \frac{n}{e}\int_0^1 x^{n-1} e^x \, dx = 1 - nI_{n-1}, \tag{2}$$

con $I_0 = 1 - 1/e$. L'applicazione diretta di questa formula ricorrente è *instabile*, cioè amplifica gli errori di arrotondamento.

Stabilità numerica - schema stabile

Dalla formula $I_n = 1 - nI_{n-1}$ possiamo ottenere il seguente schema all'indietro:

$$I_{n-1} = \frac{(1 - I_n)}{n}, \quad n = 20, 19, \ldots, 1. \tag{3}$$

Sapendo che $\lim_{n\to\infty} I_n = 0$, approssimiamo $I_n = 0$ per un n sufficientemente grande (e.g. $I_{20} = 0$). L'applicazione di questa formula è *stabile*.

Stabilità numerica - ESERCIZIO 1

IMPORTANTE: Creare una cartella denominata `LEZ03` nella home directory. Tutti gli esercizi andranno salvati nella cartella `LEZ03`.

Implementare i due schemi (stabile e instabile) per il calcolo di I_n:

$$I_n = \frac{1}{e}\int_0^1 x^n e^x \, dx, \qquad n = 0, \ldots, 20. \tag{4}$$

Per lo scopo scrivere due script distinti in MATLAB.

Stabilità numerica - SOLUZIONE ...

```
% Calcolo di n integrali con formula ricorsiva per lo studio
% della stabilita ' numerica — CASO INSTABILE
%
% Inizializzazione delle variabili
...
% Ciclo for
for ...                     % Condizione
    int = 1.0 — n * int;    % Calcolo integrale
    ...                     % Stampa a schermo le variabili n e int
end
```

ATTENZIONE: la sequenza di istruzioni all'interno del ciclo `for` vale solo per il caso instabile.

Stabilità numerica - ESERCIZIO 2

Implementare i due schemi (stabile e instabile) per il calcolo di I_n utilizzando le variabili come vettori e plottare i risultati ottenuti.

$$I_n = \frac{1}{e}\int_0^1 x^n e^x \, dx, \qquad n = 0, \ldots, 20. \tag{5}$$

Per lo scopo scrivere due script distinti in MATLAB.

Stabilità numerica - SOLUZIONE utilizzando i vettori

```
clear all
close all
% program stabile
nmax = 20;
n = nmax;
intn(n) = 0.0;
while (n > 1)
    intn(n-1) = (1.0 - intn(n)) / n;
    n = n - 1; % Aggiornamento variabile n
    fprintf('%e \n', intn(n))
end
% Plot del risultato
x = linspace(1,20,nmax);
plot(x,intn,'*')
```

ATTENZIONE: la sequenza di istruzioni all'interno del ciclo `while` vale solo per il caso stabile.

ESERCIZI

1. Calcolare in MATLAB l'espressione $((x+1)-1)/x$ per $x = 10^{-15}$. Discutere l'errore numerico commesso.
2. Valutare le espressioni equivalenti $p1(x) = (x-1)^7$ e $p2(x) = x^7 - 7x^6 + 21x^5 - 35x^4 + 35x^3 - 21x^2 + 7x - 1$ in $x = [0.99, 1.01]$ con passo $\Delta x = 0.001$ e discutere il risultato.
   Plottare il grafico delle funzioni nell'intervallo considerato.
3. Scrivere uno script per il calcolo delle radici di un'equazione di secondo grado, x_1 e x_2, e verificare il risultato nel caso in cui:
   - a,c = 1; b = 5
   - a,c = 1; b = 10^7

   Verificare la correttezza del risultato controllando che $x_1 \cdot x_2 = c/a$.
   Modificare lo script per evitare il fenomeno della cancellazione numerica:
   - Ricavare la radice x_1 non affetta da errori di cancellazione numerica (dipende dal segno di b).
   - Calcolare la seconda radice x_2 come $c/(a \cdot x_1)$.

Equazioni non lineari in una variabile

Data una funzione $f(x)$ si vogliono cercare le soluzioni del problema $f(x) = 0$ con x che varia in un certo intervallo $[a, b]$.

1. metodo di bisezione
2. metodo di punto fisso
3. metodo di Newton-Raphson
4. metodo della secante variabile

ESERCIZIO - Il metodo della bisezione in MATLAB

Implementare con uno script il metodo della bisezione per trovare la radice della funzione $f(x) = ln(x) + x^2 - sin(\pi x)$ nell'intervallo $[0, 2\pi]$.
Utilizzare lo schema seguente:

1. Partire dall'intervallo $[a_0, b_0] = [a, b]$ e calcolarne il punto medio c_0
2. Controllare i casi in cui:
   - se $f(c_0)f(a_0) > 0$ allora $[a_1, b_1] = [c_0, b_0]$
   - se $f(c_0)f(b_0) > 0$ allora $[a_1, b_1] = [a_0, c_0]$
3. La precedente procedura si ripete ogni volta dimezzando il passo dell'intervallo. Ci si ferma se l'intervallo risulta minore di una certa tolleranza imposta.

ESERCIZIO - Il metodo della bisezione in MATLAB

- Plottare la funzione $f(x)$ nell'intervallo dato e verificare che agli estremi la funzione assuma valori opposti.
- Calcolare il numero di iterazioni teorico per raggiungere la tolleranza imposta e confrontare il risultato con quello ottenuto al calcolatore.
- Stampare a schermo una tabella con i risultati ottenuti, per esempio:

NUMERO DI ITERAZIONI — ESTREMO SX — ESTREMO DX — SOLUZIONE — VALORE FX NELLA SOLUZIONE

Il metodo della bisezione - SOLUZIONE ...

Completare il codice introducendo le istruzioni mancanti al posto dei puntini

```
clear all
close all
% Script che implementa il metodo dicotomico o della bisezione
a = 0;
b = 2*pi;
toll = 10^-6;
itmax = 100;
tau = abs(a-b);
f=@(x) log(x)+x.^2-sin(pi*x);
iter = 0;
while ...
    ...
    if ...
        ...
    else
        ...
    end
    ...
    fprintf('...')
end
```