Correzione Esame di Calcolo Numerico

21/02/2019

1. La soluzione della prima espressione è

$$(3.1 \cdot 10^{200}) \cdot (2 \cdot 10^{150}) \cdot (1 \cdot 10^{-100}) = 6.2 \cdot 10^{250}\,.$$

il risultato coincide con la soluzione della seconda espressione in quanto è stata applicata semplicemente la proprietà commutativa del prodotto.
Inserendo le operazioni in Matlab si ottiene, dalla prima espressione, **Inf** e, dalla seconda espressione, il risultato corretto $6.2 \cdot 10^{250}$.
Svolgendo i calcoli da sinistra verso destra, Matlab calcola il prodotto dei primi due fattori.
Nella prima espressione

$$(3.1 \cdot 10^{200}) \cdot (2 \cdot 10^{150}) = 6.2 \cdot 10^{350} > \textbf{realmax} \simeq 1.7977 \cdot 10^{308}$$

che trattandosi di un numero $>$ **realmax** (ossia il più grande numero del sistema floating point rappresentato in Matlab) fornisce un messaggio di **overflow** rappresentato dalla variabile **Inf**.
Nella seconda espressione invece

$$(3.1 \cdot 10^{200}) \cdot (1 \cdot 10^{-100}) = 3.1 \cdot 10^{100}\,.$$

che permette di proseguire correttamente nel calcolo.

2. Risoluzione di sistemi lineari.
2a) I metodi diretti sono metodi di risoluzione che, in aritmetica esatta, forniscono la soluzione in un numero finito di passi. I metodi iterativi sono metodi per determinare una successione di approssimanti della soluzione.
2b) Dato un sistema lineare $Ax = b$ di ordine n con matrice diagonale

con matrice diagonale $A = \begin{pmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn} \end{pmatrix}$ e termine noto $b = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix}$

l'algoritmo di risoluzione è

$$x_i = \frac{b_i}{a_{ii}} \qquad i = 1, \dots, n$$

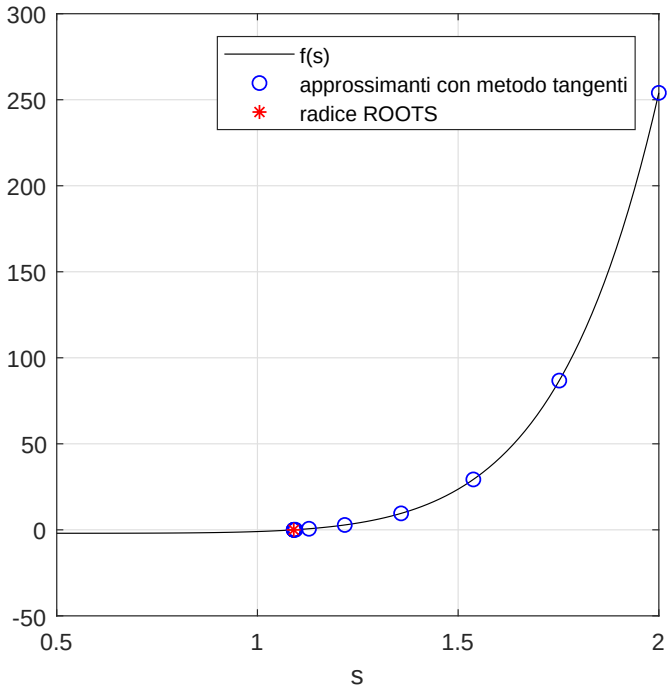
ed ha costo computazionale pari ad n divisioni.
2c) Il metodo di sostituzione all'indietro si applica iniziando a risolvere l'ultima equazione del sistema lineare $Ax = b$ con $x = (x_1, x_2, x_3)'$ e a ritroso le equazioni precedenti:

$$\begin{aligned}
&6x_3 = 12 && \rightarrow \quad x_3 = 2 \\
&4x_2 + 5x_3 = 14 && \rightarrow \quad x_2 = \frac{14 - 5x_3}{4} = \frac{14 - 5 \cdot 2}{4} = 1 \\
&1x_1 + 2x_2 + 3x_3 = \frac{59}{7} && \rightarrow \quad x_1 = \frac{59/7 - 3x_3 - 2x_2}{1} = \frac{59}{7} - 3 \cdot 2 - 2 \cdot 1 = \frac{3}{7}
\end{aligned}$$

3. I comandi Matlab necessari per lo svolgimento dei punti 3b), 3c), 3d), 3e) e 3g) sono implementati nel file **Es3.m** che richiama le funzioni **Newton.m** e **bisect.m**.
3a) Dato un valore x_0 iniziale, il metodo delle tangenti genera una successione di approssimanti della radice attraverso l'algoritmo

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \qquad k = 0, 1, \ldots$$

se $f'(x_k) \neq 0$ per $k = 0, 1, \ldots$.
Il metodo di Newton è un metodo **localmente** convergente.
3e) Il grafico risultante dallo svolgimento dei punti 3b)-3e) sarà



3f) Il metodo di Newton si arresta dopo 10 iterazioni e il valore finale differisce dall'approssimazione ottenuta con il comando **roots** di circa $4.4 \cdot 10^{-16}$.
Il metodo di bisezione si arresta dopo 35 iterazioni e il valore finale differisce dall'approssimazione ottenuta con il comando **roots** di circa $4.8 \cdot 10^{-11}$.
Il metodo di Newton risulta avere una velocità di convergenza maggiore rispetto al metodo di bisezione in accordo con le aspettative teoriche.
3g) Applicando il metodo di Newton con dato iniziale $s_0 = 0.5$, il numero di iterazioni triplica perchè $f'(0.5)$ è prossimo a 0 e quindi s_1 si allontana notevolmente dalla radice cercata. Tuttavia dopo 32 iterazioni si riottiene il valore approssimato partendo da $s_0 = 2$. Il metodo risulta quindi sensibile rispetto alla scelta del valore di innesco ma ha ancora velocità di convergenza superiore al metodo di bisezione.