Corso di Laurea in Informatica – Università di Ferrara
Calcolo Numerico
Prova scritta e pratica del 16/07/2020 – A.A. 2019–2020
Tempo a disposizione: 4 ore

Esercizio 1

1) (T) (2 punti) Determinare a quale numero reale $\alpha \in \mathbb{R}$ corrisponde la seguente rappresentazione nel formato ANSI standard IEEE a 32 bit: 11000110100101110010011001011000.

2) (M) (2 punti) Siano dati i seguenti coefficienti non nulli:

$$c_7 = \tan\left(1.37\pi^2 - \mathrm{e}^{-0.6}\right), \quad c_6 = -\log_{10}\left(4.2 + \sin\left(0.77\mathrm{e}^{1.3}\right)\right), \quad c_4 = \left|\sqrt{5.7 \cdot 10^{-2}} + \cos(-3\pi/5)\right|,$$
$$c_2 = 5\cos\left(\sin\left(12.3 - \pi^{2.1}\right)\right), \quad c_0 = \ln\left(3.1 \cdot 10^{-4}\right)$$

del polinomio $p_7(x) = c_7x^7 + \ldots + c_1x + c_0$. Realizzare un M-script file che, usando la M-function `ruffiniHorner` in allegato, calcoli e stampi il valore del polinomio $p_7(x)$ e delle sue derivate prima $p_7'(x)$ e seconda $p_7''(x)$ in un prefissato punto x_0 accettato in input. Lo script visualizzi anche il grafico del polinomio nell'intervallo $[-1.7,\ 2.0]$. Provare lo script con il valore $x_0 = 1.21$, riportando nel foglio delle risposte i valori ottenuti.

Esercizio 2

1) (T) (3 punti) Valutare l'errore inerente e l'errore algoritmico nel calcolo dell'espressione:

$$\varphi(x) = \ln\left(\frac{x-3}{4}\right) + \pi x \qquad x \in \Omega \subseteq \mathbb{R}$$

determinando l'insieme Ω di definizione. Valutare l'indice di condizionamento e l'indice algoritmico. Determinare inoltre gli eventuali valori di x per i quali il problema è mal condizionato e quelli per i quali il calcolo è instabile.

2) (M) (3 punti) Realizzare un M-function file per la valutazione della seguente funzione:

$$f(x) = -\sin\left(p_1 b^{q_1 x}\right) + \cos\left(p_2 b^{q_2 x}\right) - \sin\left(p_3 b^{q_3 x}\right) + \cos\left(p_4 b^{q_4 x}\right)$$

dove b è una costante predefinita, $\boldsymbol{p} = (p_1, \ldots, p_4)^T$ e $\boldsymbol{q} = (q_1, \ldots, q_4)^T$ sono vettori fissati. Per un fissato vettore $\boldsymbol{x}$ di valori, lo script deve eseguire i seguenti passi:

(a) calcolo di $f(x_i)$ mediante un ciclo `for` sulle componenti di $\boldsymbol{x}$;

(b) posto $y = b^x$, calcolo di tutte le $f(x_i)$ solo mediante sintassi vettoriale (*senza* alcun ciclo).

In un M-script di prova, visualizzare a video i due risultati ottenuti per ciascun x_i, utilizzando istruzioni di stampa formattate con i valori per ciascuna componente x_i di $\boldsymbol{x}$ su una stessa riga, in notazione scientifica con larghezza di campo 11 caratteri e 4 posizioni decimali. Provare lo script con $b = 0.7$, $\boldsymbol{p} = (-2, 1, 3, -1)^T$, $\boldsymbol{q} = (5, 4, 3, 2)^T$ e $\boldsymbol{x} = 1.7\left(10^{-7},\ 10^{-6},\ \ldots,\ 10^{2}\right)^T$.

Appendice: codici forniti

```
function [r, q] = ruffiniHorner(p, a)
% RUFFINIHORNER - Schema di Ruffini-Horner
%  Calcola il valore di un polinomio p(x) nel punto x = a e i coefficienti
%  del polinomio quoziente q(x) = p(x) / (x - a)
% SYNOPSIS
%   [r, q] = ruffiniHorner(p, a)
% INPUT
%   p (double array) - Vettore dei coefficienti del polinomio, ordinati
%                      da quello di grado piu' alto a quello di grado zero
%   a (double)       - Punto (numero reale) in cui calcolare il polinomio
% OUTPUT
%   r (double)       - Valore del polinomio nel punto x = a
%   q (double array) - Vettore dei coefficienti del polinomio quoziente
%                      q(x) = p(x) / (x - a)
```

```
%
r = [];
if ( isempty(p) )
    q = [];
    warning('Il vettore p dei coefficienti e'' vuoto');
    return
elseif ( isempty(a) )
    q = p;
    warning('Il punto ''a'' in cui valutare il polinomio e'' vuoto');
    return
else
    n = numel(p) - 1;  % grado del polinomio
    q = zeros(n, 1);
    q(1) = p(1);
    for i = 2 : n+1
        q(i) = q(i-1)*a + p(i);
    end
    r = q(n+1);
    q = q(1:n);
end
end % fine della function ruffiniHorner
```

Soluzioni e commenti

Nel seguito sono riportate le soluzioni degli esercizi del compito, includendo commenti che aiutino il più possibile la comprensione della soluzione. Qualora uno stesso esercizio possa eventualmente essere risolto in più modi, in alcuni casi si riportano le descrizioni anche di qualche modo alternativo. In generale, questo rende il testo delle soluzioni degli esercizi inevitabilmente molto più lungo di quanto non sia effettivamente richiesto allo studente nel momento della prova scritta, pertanto **la lunghezza delle soluzioni qui riportate è da considerarsi NON RAPPRESENTATIVA della lunghezza del compito**. Si invitano gli studenti a contattare il docente per eventuali dubbi o chiarimenti.

Nella trascrizione delle soluzioni è stata posta la massima cura. Tuttavia, nel caso si rilevino sviste e/o errori di battitura, si invitano gli studenti a comunicarle via e-mail al docente.

Soluzione esercizio 1

Teoria

$$
\begin{aligned}
\text{IEEE}_{32}(\alpha) &= 1\,\underbrace{10001101}_{\widetilde{p}}\,\underbrace{00101110010011001011000}_{\widetilde{m}} \\
s &= 1 \Rightarrow \alpha < 0 \\
\widetilde{p} &= (10001101)_2 = (+141)_{10} \\
&\Rightarrow p = \widetilde{p} - bias = 141 - 127 = 14 \\
&\Rightarrow 2^p = 2^{14} \\
\widetilde{m} &= (00101110010011001011000)_2 \\
&\Rightarrow m = (1.00101110010011001011000)_2 \\
|\alpha| &= m \cdot 2^p = (1.00101110010011001011000)_2 \cdot 2^{14} \\
&= 2^{14}\left(1 + 2^{-3} + 2^{-5} + 2^{-6} + 2^{-7} + 2^{-10} + 2^{-13} + 2^{-14} + 2^{-17} + 2^{-19} + 2^{-20}\right) \\
&= 2^{14} + 2^{11} + 2^9 + 2^8 + 2^7 + 2^4 + 2^1 + 2^0 + 2^{-3} + 2^{-5} + 2^{-6} \\
&= 16384 + 2048 + 512 + 256 + 128 + 16 + 2 + 1 + \frac{1}{8} + \frac{1}{32} + \frac{1}{64} \\
&= 19347 + 1.25 \cdot 10^{-5} + 3.125 \cdot 10^{-6} + 1.5625 \cdot 10^{-6} = 19347.171875
\end{aligned}
$$

Dunque $\alpha = (-19347.171875)_{10}$.

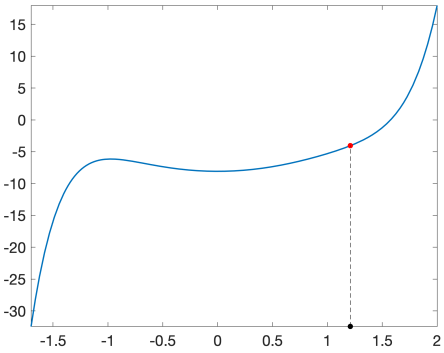
Matlab

Contenuto dell'M-script file `esercizio1.m`:

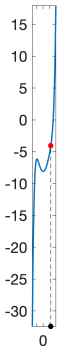
```
% Cognome Nome
% Matricola
%-----------------------------
%  esercizio 1 - 16/07/2020
%-----------------------------
close all; clear all; clc;
disp('Esecizio 1');

p7 = zeros(8,1);

p7(1) = tan( 1.37*pi^2 - exp(-0.6) );           % c7
p7(2) = -log10( 4.2 + sin(0.77*exp(1.3)) );     % c6
p7(4) = abs( sqrt( 5.7e-2 ) + cos( -3*pi/5 ) ); % c4
p7(6) = 5*cos( sin( 12.3 - pi^(2.1) ) );        % c2
p7(8) = log( 3.1e-4 );                          % c0

x0 = input('Inserire il punto nel quale valutare il polinomio (double): x0 = ');
[r, q] = ruffiniHorner( p7, x0 );
[derp, q1] = ruffiniHorner( q, x0 );
[ders, q2] = ruffiniHorner( q1, x0 );
fprintf('\nValore del polinomio in x0 = %g: p(x0) = %g',x0,r);
fprintf('\nDerivata prima del polinomio in x0: p''(%g) = %g',x0,derp);
fprintf('\nDerivata seconda del polinomio in x0: p''''(%g) = %g\n\n',x0,2*ders);
fh = fplot(@(x)(polyval(p7,x)), [-1.7,2]);
```

Eseguendo lo script si ottengono i seguenti risultati in output e una figura simile alla seguente (quella presentata qui è stata leggermente migliorata per esigenze di stampa):

Punto di valutazione: $x_0 = 1.21$
Valori calcolati in x_0:
$p_7(x_0) \approx -4.05228$
$p_7'(x_0) \approx \ \ 6.86563$
$p_7''(x_0) \approx \ 11.88157$



(a) Grafico con assi cartesiani in scale diverse (di default).



(b) Grafico con assi cartesiani nella stessa scala.

Figura 1: grafico del polinomio $p_7(x)$ dell'esercizio 1.

Nota per lo studente. Si richiama l'attenzione sul fatto che nei grafici di Matlab sia sempre MOLTO importante tenere presente in quale scala vengono visualizzati gli assi. In particolare, il motore grafico di Matlab è impostato di default nella modalità che prevede di scalare automaticamente gli assi delle ascisse e delle ordinate per far sì che l'immagine del grafico "si adatti" alle dimensioni della finestra grafica, che sono generalmente rettangolari, quasi quadrate.

Quando la scala degli assi è molto diversa, si potrebbe essere tratti in inganno dalla figura, che risulta **apparentemente** incoerente con i valori numerici delle derivate (non dei valori della funzione). Nel caso in esame, ad esempio, il valore $p_7(1.21) \approx -4.05$ è correttamente rappresentato nel grafico, ma i valori della derivata prima $p_7'(1.21) \approx 6.87$ e seconda $p_7''(1.21) \approx 11.88$ del polinomio in $x_0 = 1.21$ sembrano del tutto incoerenti con il profilo del grafico! Infatti, una derivata prima che vale quasi 7 significa che in quel punto i valori del polinomio crescono molto rapidamente, dunque nel grafico la curva dovrebbe apparire molto più ripida di quanto si veda in figura 1a. Inoltre, una derivata seconda che vale quasi 12 è indice di una forte convessità della funzione in quel punto, quindi la curva del grafico nelle vicinanze del punto dovrebbe apparire con una curvatura molto accentuata (come una "conca" piuttosto stretta), mentre in figura 1a il tratto di polinomio nelle vicinanze di $x_0 = 1.21$ appare quasi rettilineo.

Il motivo di questa discrepanza fra i valori previsti dall'analisi delle derivate e il profilo rappresentato da Matlab non è sintomo di un qualche errore, ma risiede semplicemente nel fatto che, nel caso di questo esercizio, il rapporto fra la scala s_y di visualizzazione delle ordinate e quella s_x delle ascisse è **molto grande**, ossia $s_y \gg s_x$. Senza bisogno di ispezionare le proprietà del grafico, ci si accorge molto facilmente di questa cosa semplicemente osservando l'intervallo di valori rappresentato sugli assi cartesiani: mentre le ordinate vanno da circa -32 a circa 18, le ascisse vanno da -1.7 a 2 (come richiesto dall'esercizio), su un riferimento cartesiano rettangolare con lati di lunghezza simile. In altre parole, nella figura 1a, la lunghezza ℓ_y del segmento che viene usato per rappresentare l'intervallo $[-32, 18]$ delle ordinate e la lunghezza ℓ_x del segmento usato per rappresentare l'intervallo $[-1.7, 2]$ delle ascisse sono molto simili: $\ell_x \approx 1.2\ell_y$. È chiaro quindi che la scala s_y delle ordinate debba essere molto più grande di quella delle ascisse, per poter rappresentare in segmenti di lunghezza simile un intervallo ampio circa $18 - (-32) = 40$ per le ordinate, a fronte di un intervallo ampio $2 - (-1.7) = 3.7$ per le ascisse. In questo caso, dunque, il rapporto s_y/s_x fra le scale di figura 1a risulta circa $(40/\ell_y)/(3.7/\ell_x) = (40 \cdot 1.2)/3.7 \approx 13$ (quello effettivo vale circa 13.6, ricavabile dalla proprietà `DataAspectRatio` del *graphics handle* della figura stessa). Ciò significa, essenzialmente, che per avere la curva nelle reali proporzioni dovremmo comprimere di circa 13 volte l'asse delle ascisse, mantenendo fissato quello delle ordinate. Tutto questo, in Matlab, si ottiene direttamente impostando una proprietà degli assi, attraverso il comando `axsis('equal')`, che fa sì che vengano (ri)disegnati con la stessa scala. Ciò che si ottiene è il grafico nella figura 1b. Qui i due intervalli delle ascisse e delle ordinate sono nella corretta proporzione: quello delle ordinate occupa un segmento lungo oltre 13 volte quello occupato dall'intervallo delle ascisse. Con questa visualizzazione, allora, possiamo facilmente notare come la curva di $p_7(x)$ nelle vicinanze di $x_0 = 1.21$ sia effettivamente crescente in modo molto "ripido" e la convessità sia molto accentuata, in perfetta coerenza con quanto predetto dai valori numerici delle derivate in x_0.