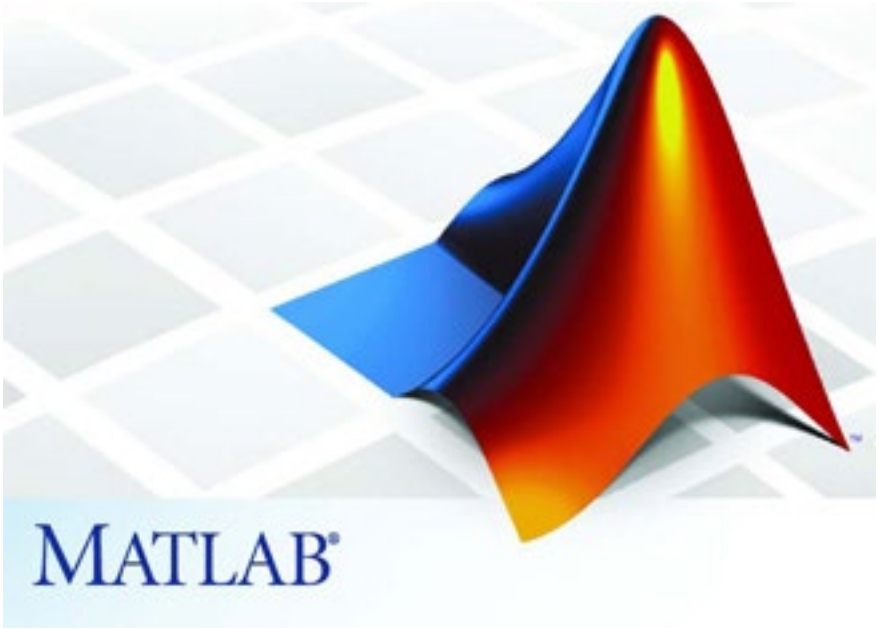
ISSN 2239-7035

dicecca.net - Computer Science
Anno II - № 21 – Supplemento al MONITORE NAPOLETANO - № 163 – Settembre 2021

GIOVANNI DI CECCA

Matematica Applicata E Computazionale

Esercizi di



dicecca.net
web site

www.monitorenapoletano.it/dicecca.net-cs

Direttore Responsabile: Giovanni Di Cecca

Collana dicecca.net – Computer Science

Anno II - № 21 – Supplemento al Numero 163 – Settembre 2021

Periodico Mensile Registrato presso il Tribunale di Napoli № 45 dell'8 giugno 2011

ISSN: 2239-7035

Introduzione

Il seguente elaborato contiene lo svolgimento degli esercizi che sono stati proposti.

Gli Elaborati sono suddivisi in 3 Livelli di Difficoltà, più alcuni Quiz.

Dove è stato necessario si è considerato lo scorporamento della funzione in più sotto funzioni chiamate da un main.

Questa soluzione è stata importante in quanto alcuni esercizi necessitavano di chiamare la stessa funzione, e quindi, da un punto di vista ingegneristico si è fatto un uso del riciclo di codice.

Di seguito viene riportata la lista di tutti i numeri degli esercizi con indicazione del livello di difficoltà di afferenza.

Tale lista fa riferimento al **programma dell'a. a. 2014/2015.**

I numeri sottolineati corrispondono agli esercizi svolti

Livelli di Difficoltà

Liv 1:(obbligatori) 1a, 2, 3, 4, 5, 6a, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 28, 29, 31, 37, 47, 51

Liv 2: 6b, 21, 24, 25, 30, 39

Liv 3: 1b

Quiz: 10, 20, 21, 22, 33, 41

Elaborati Svolti

Livello 1: 22 (-1)

Livello 2: 6

Livello 3: 1

Quiz: 6

Numeri Complessi

Es.1 – Liv. 1 – Il Tetris

Il famoso gioco creato nel 1984 da Alexey Pažitnov, Vadim Gerasimov e Dmitrij Pavlovskij, al netto delle implicazioni computazionali sulla possibilità di vittoria del giocatore, basa la sua struttura su una soluzione di numeri complessi che danno forma ai blocchetti mediante l'uso di array:

```
F1=[0+Q 1+Q i+Q 1+i+Q];             % Quadrato
F2=[0+Q 1+Q i+Q 2i+Q];              % L capovolta
F3=[0+Q 1+2i+Q i+Q 2i+Q];           % Forma L
F4=[0+Q 1+Q 1+i+Q 2+i+Q];           % Forma z
F5=[-1+Q -1+i+Q -1+2*i+Q -1+3*i+Q]; % Asta
```

La prima versione proposta non usa dei semplici controlli di "girata", mentre la seconda versione usa il toolbox `Uicontrol()`

```
%   Matematica Applicata e Computazionale
%
%   Tetris - Liv 1
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
% USO
%
% Lanciare da consolle il file
%
% >> tetris_liv_1
%
%
```

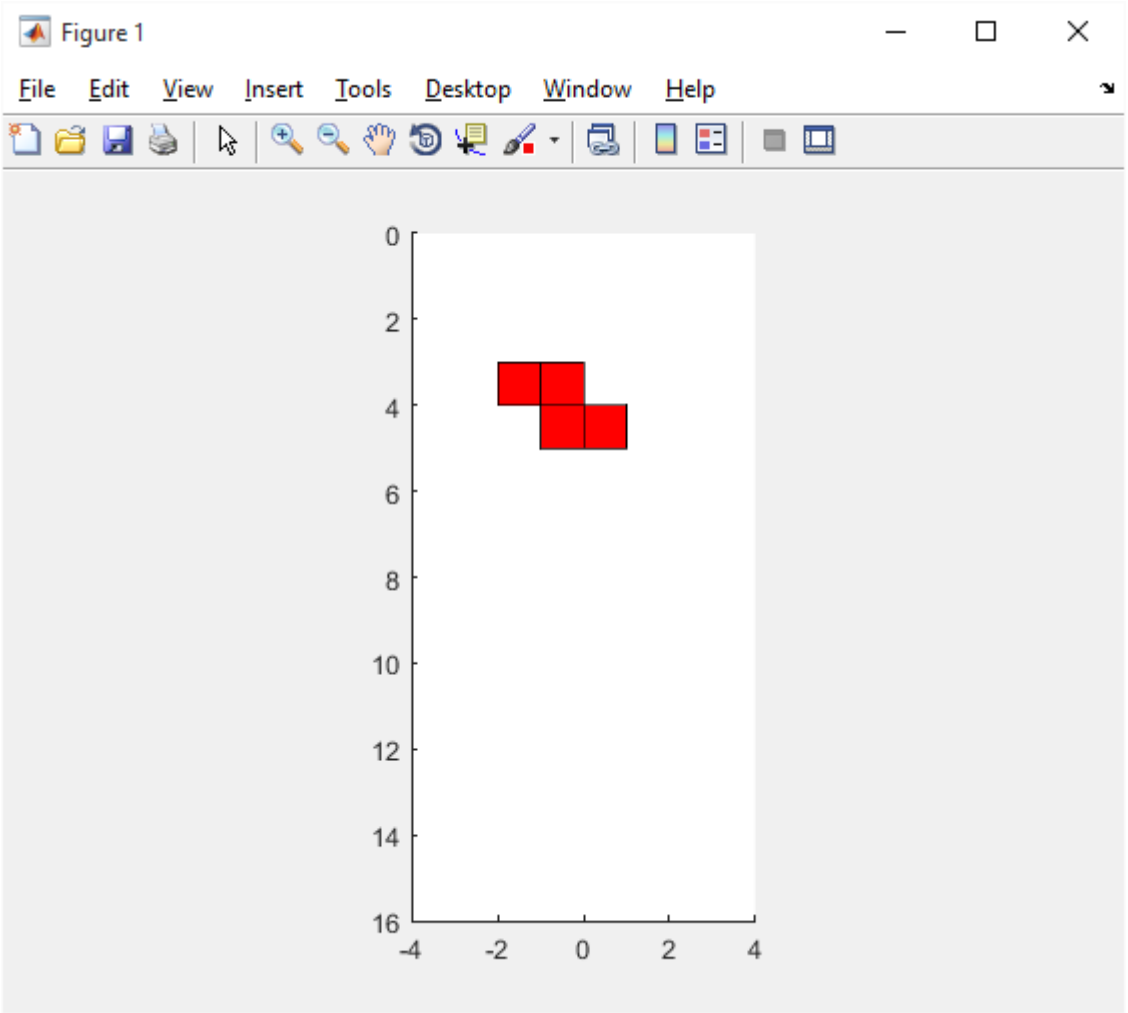
```
clc % Pulisce la schermata di MATLAB

% Crea la finestra grafica dove scenderà la forma
% e la griglia viene creata per poter vedere le coordinate dei punti
minX=-4; maxX=4; minY=0; maxY=16; % Variabili che creano la dim degli
assi
V=[minX maxX minY maxY]; % Vettore che contiene gli assi

% cambio sistema di riferimento.Il sistema ij viene usato per
% immagini e matrici.L'origine si sposta in alto
axis equal; axis(V);
axis ij;
grid % crea la griglia

% quadratino di base con cui componiamo le forme del tetris
Q=[0 1 1+i i 0 ].';% un vertice è in 0 e il lato è unitario

% Forme del Tetris: ricordiamo che il sistema di riferimento è ij
F1=[0+Q 1+Q i+Q 1+i+Q];             % Quadrato
F2=[0+Q 1+Q i+Q 2i+Q];              % L capovolta
F3=[0+Q 1+2i+Q i+Q 2i+Q];           % Forma L
F4=[0+Q 1+Q 1+i+Q 2+i+Q];           % Forma z
F5=[-1+Q -1+i+Q -1+2*i+Q -1+3*i+Q]; % Asta
numforme = 5;                       % Numero complessivo di forme

% centra orizzontalmente rispetto allo 0
F1 = F1 - 1;
F2 = F2 - .5;
F3 = F3 - .5;
F4 = F4 -1.5;
F5 = F5 + .5;

% Scegli forma a caso
% la variabile Forme è un array tridimensionale che ha 6 righe,
% 4 colonne ( ovvero la taglia di F1 F2 F3 F4 F5 ) ed in più ha 5 facce,
% una per ogni forma predefinita
Forme = cat (3,F1,F2,F3,F4,F5);

% ottiene un valore casuale associato ad una forma e ad un colore
j=1+round(rand*(numforme-1));
F=Forme(:,:,j); % Sceglie la forma associata all'indice ottenuto
% array colori possibili tra cui scegliere:
```

```
% (magenta, nero, blu, rosso, giallo, ciano)
color = ['m','k','b','r','y','c'];

patch(real(F),imag(F),color(j)) % disegna la forma del poligono
% discesa della forma e scelta casuale di una tra le seguenti azioni:
% 1-sposta a sinistra
% 2-sposta a destra
% 3-ruota di -90° o + 90°
while all(all(imag(F) < maxY))
    pause(1) % velocità di discesa, ogni 1 sec
    F=F+i;
    clf % Clear current figure window - Pulisce lo schermo ad ogni
passaggio
    patch(real(F),imag(F),'g')
    axis equal; axis([-4 4 0 16])
    axis ij

    % Calcolo del baricentro e della nuova forma ruotata.
    % Per evitare che la rotazione avvenga attorno all'origine,
    % calcoliamo il baricentro della forma. La forma generata è una
matrice
    % di 5 righe e 4 colonne, ogni colonna ha un quadratino di base.
    % E' possibile calcolare, quindi, i baricentri dei singoli quadratini
    % e passare poi al calcolo del baricentro dei baricentri.
    % Baricentro di ogni quadratino: somma delle coordinate di ogni punto
    % diviso il numero dei vertici del singolo quadratino
    bar = sum (F((1:end-1), :))/4;

    % Baricentro dei baricentri: somma dei singoli baricentri diviso il
numero di quadratini
    baricentro = sum(bar)/4;

    % Per eseguire correttamente la rotazione, bisogna prima traslare
    % l'origine del sistema di riferimento nel baricentro della figura.
    rot90 = F-baricentro;     % Coordinate attuali - Coord nuova origine
    rot90 = i*rot90;          % Rotazione di 90°
    rot90 = rot90+baricentro; % Riportiamo il sistema di riferimento
nella posizione originaria

    % Rotazione di -90°
    rotmeno90 = F-baricentro; % Coordinate attuali - Coord nuova origine
    rotmeno90 = rotmeno90/i;  % Rotazione di -90°
```

```
    rotmeno90 = rotmeno90+baricentro; % Riportiamo il sistema di
riferimento nella posizione originaria

    % SCELTA DELLE POSSIBILI AZIONI DA ESEGUIRE IN BASE ALLA POSIZIONE
    % DELLA FIGURA.
    % 1-Se la figura non si trova sui bordi ...
    % all determina se gli elementi degli array siano diversi da 0
    if (all(all(real(F)>minX)) && all(all(real(F)<maxX)))
        % .. è possibile eseguire tutte le azioni.
        % mov è un vettore tridimensionale formato da 5 righe, 4 colonne
        % e 4 facce, ogni faccia contiene i valori dello spostamento
laterale
        % o della rotazione della j-sima figura F
        mov = cat(3, F-0.5, F+0.5, rot90, rotmeno90);

        %Generazione di interi casuali tra 1 e 4
        k = 1+round(rand*3);
        F = mov(:, :, k); % Viene scelto il k-simo movimento

    % 2-Se la figura si trova sul bordo sinistro sono possibili tre
azioni:
    % spostamento a destra e rotazione di 90° e -90°.
    elseif all(all(real(F)==minX))

        % mov è un vettore tridimensionale formato da 5 righe, 3 colonne
        % e 3 facce, ogni faccia contiene i valori dello spostamento
        % laterale o della rotazione della j-sima figura F
        mov = cat(3, F+0.5, rot90, rotmeno90);

        % Generazione di interi casuali da 1 a 3
        k = 1+round(rand*2);
        F = mov(:, :, k); % Viene scelto il k-simo movimento

    % 3-Se la figura si trova sul bordo destro sono possibili tre azioni:
    % spostamento a sinistra e rotazione di 90° e -90°.
    else

        % mov è un vettore tridimensionale formato da 5 righe, 3 colonne
        % e 3 facce, ogni faccia
        % contenente i valori dello spostamento laterale o della
        % rotazione della j-sima figura F
        mov = cat(3, F-0.5, rot90, rotmeno90);
```

```
        % Generazione di interi casuali tra 1 e 3
        k = 1+round(rand*2);
        F = mov(:, :, k); % Viene scelto il k-simo movimento

    end % Chiudi IF

    clf % Pulisci figura

    patch(real(F), imag(F), color(j)); % Disegna la figura F dopo il
movimento
    axis equal; axis(V); % Disegna la finestra grafica
    axis ij;

end % chiudi while
```

Esempio d'uso



Es.1 – Liv. 3 – Il Tetris

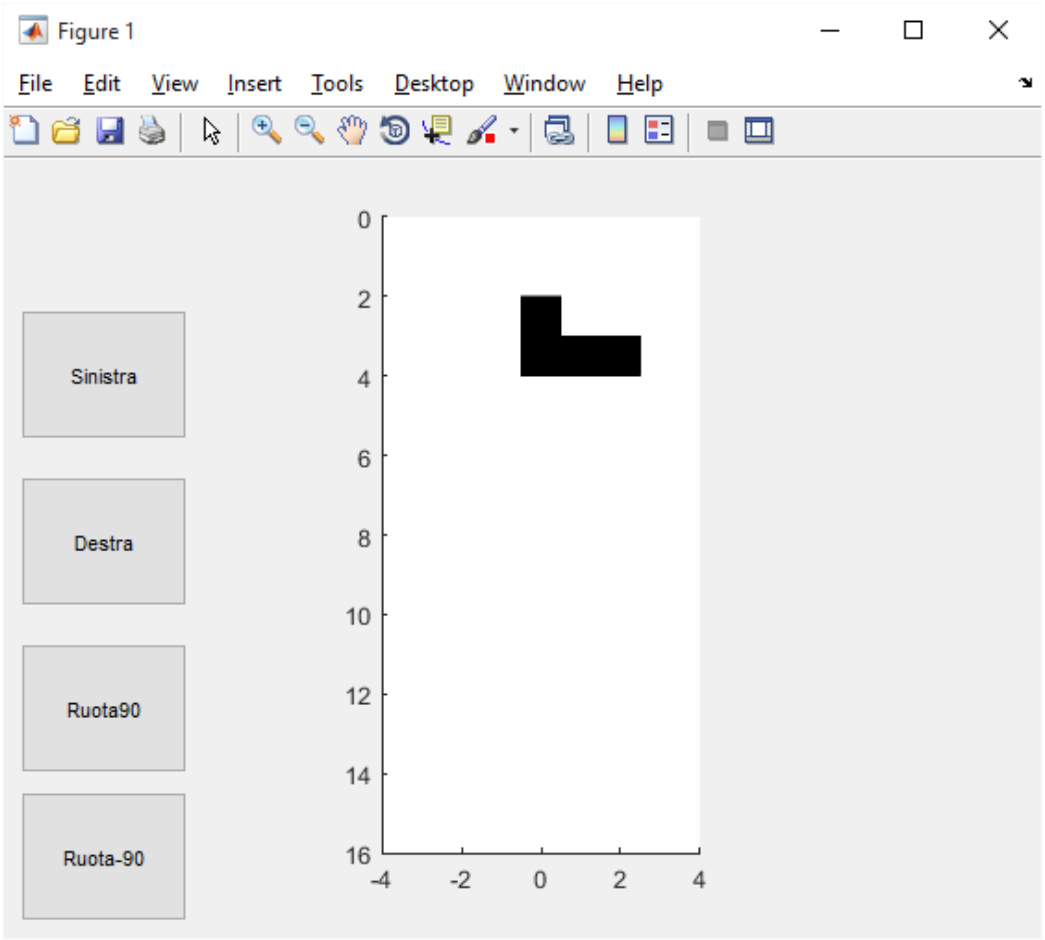
```
%
%   Matematica Applica e Computazionale
%
%   Tetris - Liv 3
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
% USO
%
% Lanciare da consolle il file
%
% >> tetris_liv_3
%
% Cliccare i pulsanti:
%
% Sinistra
% Destra
% Ruota 90
% Ruota -90

clc % Pulisce la schermata di MATLAB

% Crea la finestra grafica dove scenderà la forma
% e la griglia viene creata per poter vedere le coordinate dei punti
minX=-4; maxX=4; minY=0; maxY=16; % Variabili che creano la dim degli
assi
V=[minX maxX minY maxY]; % Vettore che contiene gli assi

% cambio sistema di riferimento.Il sistema ij viene usato per
```

```
% immagini e matrici.L'origine si sposta in alto
axis equal; axis(V);
axis ij;
grid % crea la griglia

% quadratino di base con cui componiamo le forme del tetris
Q=[0 1 1+i i 0 ].';% un vertice è in 0 e il lato è unitario

% Forme del Tetris: ricordiamo che il sistema di riferimento è ij
F1=[0+Q 1+Q i+Q 1+i+Q]; % Quadrato
F2=[0+Q 1+Q i+Q 2i+Q]; % L capovolta
F3=[0+Q 1+2i+Q i+Q 2i+Q]; % Forma L
F4=[0+Q 1+Q 1+i+Q 2+i+Q]; % Forma z
F5=[-1+Q -1+i+Q -1+2*i+Q -1+3*i+Q]; % Asta
numforme = 5; % Numero complessivo di forme

% centra orizzontalmente rispetto allo 0
F1 = F1 - 1;
F2 = F2 - .5;
F3 = F3 - .5;
F4 = F4 -1.5;
F5 = F5 + .5;

% Scegli forma a caso
% la variabile Forme è un array tridimensionale che ha 6 righe,
% 4 colonne ( ovvero la taglia di F1 F2 F3 F4 F5 ) ed in più ha 5 facce,
% una per ogni forma predefinita
Forme = cat (3,F1,F2,F3,F4,F5);

% ottiene un valore casuale associato ad una forma e ad un colore
j=1+round(rand*(numforme-1));
F=Forme(:,:,j); % Sceglie la forma associata all'indice ottenuto

% array colori possibili tra cui scegliere:
% (magenta, nero, blu, rosso, giallo, ciano)
color = ['m','k','b','r','y','c'];

patch(real(F),imag(F),color(j)) % disegna la forma del poligono
% discesa della forma e scelta casuale di una tra le seguenti azioni:
% 1-sposta a sinistra
% 2-sposta a destra
% 3-ruota di -90° o + 90°
```

```
while all(all(imag(F) < maxY))
    pause(1) % velocità di discesa, ogni 1 sec
    F=F+i;
    clf % Clear current figure window - Pulisce lo schermo ad ogni
passaggio
    patch(real(F),imag(F),'g')
    axis equal; axis([-4 4 0 16])
    axis ij

    % Calcolo del baricentro e della nuova forma ruotata.
    % Per evitare che la rotazione avvenga attorno all'origine,
    % calcoliamo il baricentro della forma. La forma è una matrice
    % di 5 righe e 4 colonne, ogni colonna ha un quadratino di base.
    % E' possibile calcolare, quindi, i baricentri dei singoli quadratini
    % e passare poi al calcolo del baricentro dei baricentri.
    % Baricentro di ogni quadratino: somma delle coordinate di ogni punto
    % diviso il numero dei vertici del singolo quadratino
    bar = sum (F((1:end-1), :))/4;

    % Baricentro dei baricentri: somma dei singoli baricentri diviso il
    % numero di quadratini
    baricentro = sum(bar)/4;

    % Per eseguire correttamente la rotazione, bisogna prima traslare
    % l'origine del sistema di riferimento nel baricentro della figura.
    rot90 = F-baricentro; % Coordinate attuali - Coord nuova origine
    rot90 = i*rot90; % Rotazione di 90°
    rot90 = rot90+baricentro; % Riportiamo il sistema di riferimento
                              % nella posizione originaria

    % Rotazione di -90°
    rotmeno90 = F-baricentro; % Coordinate attuali - Coord nuova origine
    rotmeno90 = rotmeno90/i; % Rotazione di -90°
    rotmeno90 = rotmeno90+baricentro; % Riportiamo il sistema di
                                % riferimento nella posizione originaria

    % SCELTA DELLE POSSIBILI AZIONI DA ESEGUIRE IN BASE ALLA POSIZIONE
    % DELLA FIGURA.
    % 1-Se la figura non si trova sui bordi ...
    % all determina se gli elementi degli array siano diversi da 0
    if (all(all(real(F)>minX)) && all(all(real(F)<maxX)))
        % .. è possibile eseguire tutte le azioni.
```

```
        % mov è un vettore tridimensionale formato da 5 righe, 4 colonne
        % e 4 facce, ogni faccia contiene i valori dello spostamento
        % laterale o della rotazione della j-sima figura F
        mov = cat(3, F-0.5, F+0.5, rot90, rotmeno90);

        % Generazione di interi casuali tra 1 e 4
        k = 1+round(rand*3);
        F = mov(:, :, k); % Viene scelto il k-simo movimento

    % 2-Se la figura si trova sul bordo sinistro sono possibili tre
    % azioni spostamento a destra e rotazione di 90° e -90°.
    elseif all(all(real(F)==minX))

        % mov è un vettore tridimensionale formato da 5 righe, 3 colonne
        % e 3 facce, ogni faccia contiene i valori dello spostamento
        % laterale o della rotazione della j-sima figura F
        mov = cat(3, F+0.5, rot90, rotmeno90);

        % Generazione di interi casuali da 1 a 3
        k = 1+round(rand*2);
        F = mov(:, :, k); % Viene scelto il k-simo movimento

    % 3-Se la figura si trova sul bordo destro sono possibili tre azioni:
    % spostamento a sinistra e rotazione di 90° e -90°.
    else
        % mov è un vettore tridimensionale formato da 5 righe, 3 colonne
        % e 3 facce, ogni faccia contenente i valori dello spostamento
        % laterale o della rotazione della j-sima figura F
        mov = cat(3, F-0.5, rot90, rotmeno90);

        % Generazione di interi casuali tra 1 e 3
        k = 1+round(rand*2);
        F = mov(:, :, k); % Viene scelto il k-simo movimento

    end % Chiudi IF

    clf % Pulisci figura

    patch(real(F), imag(F), color(j)); % Disegna la figura F dopo il
                                       % movimento
    axis equal; axis(V); % Disegna la finestra grafica
    axis ij;
```

```
    %%%%%%%%%%%%%%%%%%%%%%%%
    % PULSANTI DIREZIONALI %
    %%%%%%%%%%%%%%%%%%%%%%%%
    Sinistra=uicontrol('Position',[10 270 90
70],'String','Sinistra','Callback','eval(''F=F-1;'')');
    Destra=uicontrol('Position',[10 180 90
70],'String','Destra','Callback','eval(''F=F+1;'')');

    % Ruota 90
    Ruota=uicontrol('Position',[10 90 90
70],'String','Ruota90','Callback','eval(''rot90=F-baricentro;
rot90=i*rot90; rot90=rot90+baricentro;'')');
    % Ruota -90
    Ruota2=uicontrol('Position',[10 10 90 70],'String','Ruota-
90','Callback','eval(''rotmeno90=F-baricentro; rotmeno90=rotmeno90/i;
rotmeno90=rotmeno90+baricentro;'')');
    hold on % Poiché ' clf cancella anche i bottoni disegnati ad ogni
iterazione
end % chiudi while
```



Radici di Numeri Complessi

ES. 2 – Liv. 1 - Radici Ennesime

Scrivere una *function* (e controllarne i risultati) che, dati in input un numero complesso `z` ed un intero positivo `n`, restituisca tutte le radici `n`sime di `z` ottenute mediante `roots()` con il medesimo ordinamento delle radici ottenute con la formula che fa uso delle coordinate polari

$$w_k = \sqrt[n]{z} = [r, \varphi_k] = \begin{cases} r = \sqrt[n]{\rho} \\ \varphi_k = \dfrac{\theta}{n} \pm \dfrac{2k\pi}{n}, \end{cases} \qquad k = 0, 1, \ldots, n-1$$

Il problema è stato scisso in due function: un main e la funzione vera e propria che calcola l'esercizio

```
%
%   Matematica Applicata e Computazionale
%
%   Radici Ennesime - Liv 1
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
% USO
%
% Lanciare da consolle il file
%
% >> radicinsime
```

```
%
% Inserire per z = 4+2i
% Inserire per n = 5
%
% e vedere i calcoli che esegue

% Questo esercizio prevede l'uso di due funzioni una
% principale che ha un input da tastiera ed una funzione vera
% e propria che fa il calcolo

% %%%%
% Main
% %%%%

clc % Pulisci la schermmata MATLAB

% Inserisci i dati da tastiera
z=input('Inserire un Numero Complesso z = ');
n=input('Inserire Grado della Radice n = ');

% Carica la funzione che ha per nome file f_radicinsime.m
% all'interno della cartella
[radici_ennesime]=f_radicinsime(z,n);
```

```
%
%   Matematica Applicata e Computazionale
%
%   Radici Ennesime - Liv 1
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%

% %%%%%%%%%%%%%%%%%%%%%%%%%
% Funzione f_radicinsime.m
% %%%%%%%%%%%%%%%%%%%%%%%%%
function [radici_ennesime]=f_radicinsime(z,n)

% calcolo delle radici n-sime di z usando la funzione ROOTS()
w=roots([1 zeros(1,n-1) -z]);

% Attiva la parte grafica
display('Radici roots()')
disp(w)
figure
h=compass(w,'r'); % Radici calcolati da roots in Rosso (r)
set(h,'LineWidth',4);
hold on % Blocca

% CALCOLO DELLE RADICI n-esime (radici_mat)di z usando la
% formula matematica.
%
% Le coordinate di una radice cosi calcolata sono:
% R (modulo) e phi (argomento)
% dove
% R=|z|^(1/n)
% phi=(theta/n)+(2kpi/n)
```

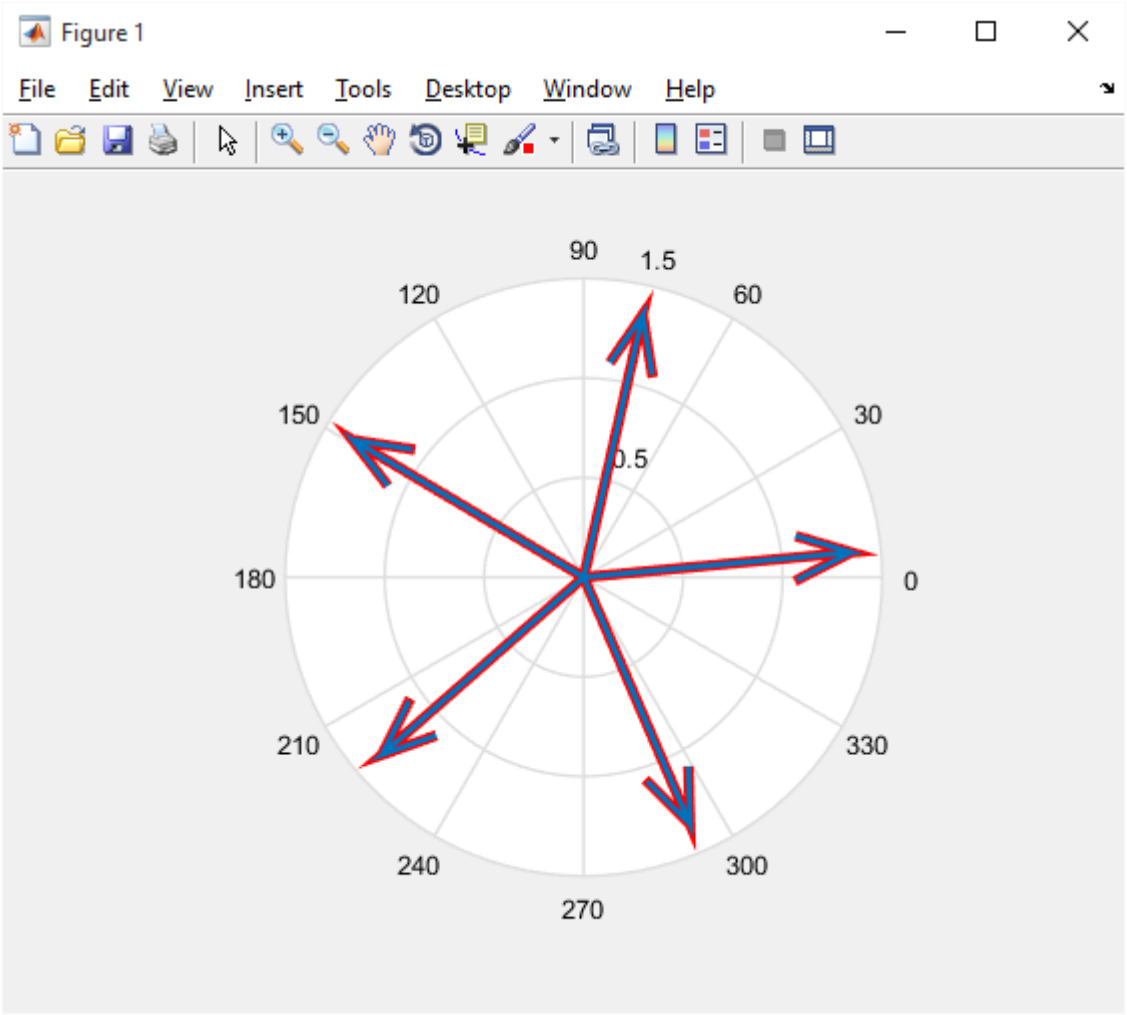
```
%
% TALI RADICI HANNO UN ORDINE DIVERSO DA roots()
R=abs(z)^(1/n);%|z|^(1/n)

k=(0:n-1)';

phi=angle(z)/n+2*k*pi/n; % phi=(theta/n)+(2kpi/n)

w_mat=R*exp(1i*phi);

% Scrivi nella schermata grafica
display('Radici con Formula Matematica')
disp(w_mat)
h=compass(w_mat); % DISEGNO RADICI DELLA FORMULA MATEMATICA w_mat
set(h,'LineWidth',2);
hold on

% Le radici ottenute con la formula matematica (w_mat)
% sono ordinate nell'intervallo trigonometrico [0,2pi] mentre quelle di
% roots() seguono l'intervallo degli argomenti principali  [-pi,pi]
% Per ottenere il medesimo ordine occorre riordinare gli argomenti.

% Useremo il comando
% angle([w_mat w]);

% Applicando SORT sugli argomenti ottenuti con ROOTS () essi verranno
% ordinati in senso crescente.Le variabili usate sono:
% - w_sort: vettore con gli argomenti di roots ordinati in senso
crescente
% - jj: indice indicante la loro posizione nel vettore originario
ottenuto con roots()
% - angle(w):vettore originario argomenti delle radici ottenute con
roots( )
[w_sort,jj]=sort(angle(w));
disp('Indice indicante la loro posizione nel vettore originario ottenuto
con roots()')
disp(jj)

% Trovo l'indice f di posizione del primo angolo maggiore di 0
% per ruotare con  circshift() il vettore di numero di posizioni corretto
f=find(angle(w(jj))>0,1,'first');
```

```
% Rotazione degli argomenti circshift()
angle([w_mat,circshift(w(jj),(f-1))]);

% Rotazione delle radici complesse circshift()
radici_ennesime=[w_mat,circshift(w(jj),(f-1))];

disp ('Radici Ennesime');
disp (radici_ennesime); % stampa i valori delle radici ennesime

% ruota il vettore di numero di posizioni corretto
display('Radici Formula - Radici Roots ordinate – Figura')

end % fine funzione
```

Esempio d'uso

```
Inserire un Numero Complesso z = 4+2i
Inserire Grado della Radice n = 5
Radici roots()
  -1.1603 + 0.6886i
  -1.0135 - 0.8908i
   0.2963 + 1.3163i
   0.5340 - 1.2391i
   1.3435 + 0.1249i

Radici con Formula Matematica
   1.3435 + 0.1249i
   0.2963 + 1.3163i
  -1.1603 + 0.6886i
  -1.0135 - 0.8908i
   0.5340 - 1.2391i

Indice indicante la loro posizione nel vettore originario ottenuto con
roots()
     2
     4
     5
     3
     1
```

```
Radici Ennesime
   1.3435 + 0.1249i   0.2963 + 1.3163i
   0.2963 + 1.3163i  -1.1603 + 0.6886i
  -1.1603 + 0.6886i  -1.0135 - 0.8908i
  -1.0135 - 0.8908i   0.5340 - 1.2391i
   0.5340 - 1.2391i   1.3435 + 0.1249i

Radici Formula - Radici Roots ordinate - Figura
```



ES. 3 – Liv. 1 – Somma Prodotto

Detto `r` l'array delle radici `n`-sime del numero complesso `z`, perché risulta sempre

(∀`n, ∀z`) `sum(r)==0` e `prod(r)==(-1)^(n-1)*z`?

Il Problema e la soluzione

Immaginiamo di voler calcolare le radici n-sime di un numero complesso z, con n = 4:

$$r = z^{\frac{1}{4}} = (2+i)^{\frac{1}{4}}$$

Elevando entrambi i membri alla potenza n-sima e portando tutto al primo membro, otteniamo:

$r^4 = 2 + i$

$r^4 = (2 + i) = 0$

A questo punto, abbiamo ottenuto un'equazione con un polinomio di grado 4. In un polinomio di grado n abbiamo sempre n+1 coefficienti (quindi, in questo caso, 5) e il coefficiente della potenza di grado massimo è pari a 1, i coefficienti intermedi pari a 0 e il termine noto uguale a -z, ove z è il numero complesso.

$$1 \times r^4 + 0 \times r^3 + 0 \times r^2 + 0 \times r^1 - (2 + i) = 0$$

Risolvere l'equazione significa trovare le radici del polinomio.

Introduciamo le FORMULE DI VIETE, che mettono in relazione le radici di un polinomio con i suoi coefficienti. TALI FORMULE AFFERMANO CHE :

"Se $P(X) = a_n * X^n + a_{(n-1)} * X^{(n-1)} + \cdots + a_1 * X + a_0$

è un polinomio di grado n >= 1 a

coefficienti complessi (cioè a_0, a_1, a_{n-1}, a_n sono NUMERI complessi, con a_n ≠0), per il

Teorema Fondamentale dell'Algebra ….

P(X) ha n radici complesse (non necessariamente distinte): $x_1, x_2, \ldots, x_n$"

In particolare, le formule di Viète affermano che:

1) $x_1 + x_2 + \cdots + x_n = -\left(\frac{a_n - 1}{a\_n}\right)$

2) $x_1 * x_2 * \ldots * x_n = -1^n * \frac{a_0}{a_n}$

Nel nostro caso a_{n-1} e a_n valgono sempre 0 e 1 e, quindi, grazie alla (1), possiamo concludere

che la somma delle n radici n-sime di un numero complesso vale sempre 0.

Analizzando la (2), invece, possiamo dire che il rapporto per a_n non influisce, dal momento che

a_n è uguale a 1.

Il termine a_0, invece, rappresenta il termine noto (-z) che, moltiplicato per -1^n, comporta due possibili risultati, in base al segno.

In particolare, effettuando il prodotto fra le n radici n-sime di z, otterremo -z se n è pari, +z se n è dispari.

Anche in questo caso abbiamo suddiviso il tutto in due function

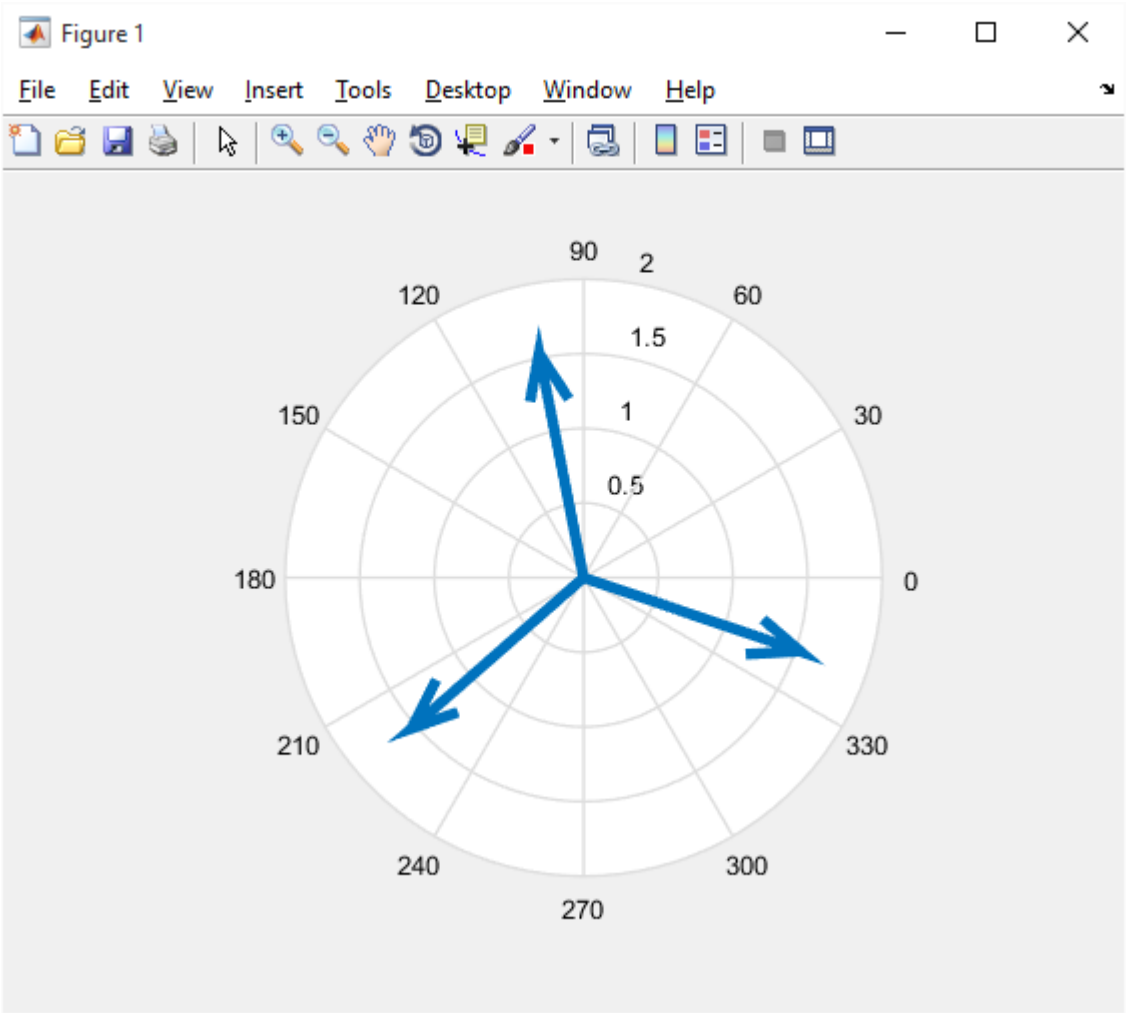
```
%   Matematica Applicata e Computazionale
%
%   Somma-Prodotto - Liv 1
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
% USO
%
% Lanciare da consolle il file
%
% >> sommaprodotto
%
% Numero Complesso nella forma algebrica a+ib = 2+3i
% Inserire Ordine della Radice n-esima: 3
%
% e vedere i calcoli che esegue

% Questo esercizio prevede l'uso di due funzioni una
% principale che ha un input da tastiera ed una funzione vera
% e propria che fa il calcolo

% %%%%
% Main
% %%%%

clc % Pulisci la schermmata MATLAB

% Inserimento dati in input
z=input('Inserire Numero Complesso nella forma algebrica a+ib: ');
n=input('Inserire Ordine della Radice n-esima: ');
```

```
% Chiamata alla funzione che effettua materialmente somma e prodotto
[somma,prodotto] = f_sommprod(z,n);

% Visualizzazione dei risultati
display(['Somma=' num2str(somma) ' --- Prodotto= ' num2str(prodotto)]);
```

```
%   Matematica Applicata e Computazionale
%
%   Somma-Prodotto - Liv 1
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%  GNU/GPL License
%

% %%%%%%%%%%%%%%%%%%%%%%%
% Funzione: f_sommprod.m
% %%%%%%%%%%%%%%%%%%%%%%%
function [somma,prodotto] = f_sommprod(z,n)

radici=roots([1 zeros(1,n-1) -z])
% AVREMMO POTUTO CALCOLARE LE RADICI n-esime ANCHE
% ATTRAVERO IL SEGUENTE CALCOLO DELLE COORDINATE
% ESPONENZIALI DI UNA RADICE
% APPLICANDO POI AD ESSE LA FORMULA ESPONENZIALE
%
% R=abs(z)^(1/n);
% k=(0:n-1)';
% phi=angle(z)/n+2*k*pi/n;
% radici_mat=R*exp(1i*phi);
% somma=sum(radici);
somma=sum(radici);
prodotto = prod(radici); % IL PRODOTTO FRA LORO E' UGUALE AL NUMERO
                         % COMPLESSO STESSO
prodotto = -(prod(radici)); % DOBBIAMO OPERARE UNA OPPORTUNA CORREZIONE
                            % DELLA POSITIVITA'
h=compass(radici);
set(h,'LineWidth',4);

end % END Function
```

Esempio D'Uso

```
Inserire Numero Complesso nella forma algebrica a+ib: 2-3i
Inserire Ordine della Radice n-esima: 3

radici =

  -1.1532 - 1.0106i
  -0.2986 + 1.5040i
   1.4519 - 0.4934i

Somma=6.6613e-16 --- Prodotto= -2+3i
```



ES. 4 – Liv. 1 – Grado di Radici complesse

In funzione dei parametri *n* e *z*, visualizzare le radici $\sqrt[n]{z}$ mediante le funzioni grafiche 3D `(ezsurf(), ezmesh(),…)` del *Symbolic Math Toolbox*.

```
%
%   Matematica Applicata e Computazionale
%
%   gcomplexrad - Liv 1
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
% USO
%
% Lanciare da consolle il file
%
% >> gcomplexrad
%
% Inserire un Numero Complesso z (nella forma a+ib) = 2+3i
% Inserire Grado della Radice n = 3
%
% e vedere i calcoli che esegue

clc % Pulisci la schermmata MATLAB

% Inserimento valori di input
z=input('Inserire un Numero Complesso z (nella forma a+ib) = ');
n=input('Inserire Grado della Radice n = ');
syms a b real;
```

```
% POSTA LA CONDIZIONE
%
% w^n=z <=> w=z^(1/n)
%
% LA RADICE w E' UN NUMERO COMPLESSO DEFINITO DA w=a+ib
% E POICHE' LE FUNZIONI ezsurf() ED ezmesh() PRENDONO IN INPUT UNA
FUNZIONE
% DEFINIREMO UNA F CHE DESCRIVA LA CONDIZIONE w^n=z
% SI OTTIENE ALGEBRICAMENTE (w^n)-z=0
% QUINDI f RISULTA
% f=(w^n)-z
% CALCOLO DELLA FUNZIONE F CHE ESPRIME LA RELAZIONE FRA UN COMPLESSO z E
% LA SUA RADICE w
w=a+1i*b;
F=(w^n)-z;
% disegna il modulo della funzione simbolica rappresentante le radici
% n-sime di z prima con ezsurf ( che traccia una superfice 3D) poi con
% ezmesh (griglia 3D )
figure % figure1
ezsurf(abs(F), [-1.2 1.2]); % DISEGNO DEL MODULO DI f IN 3D con ezsurf()
title('Funzione ezsurf( ) su Modulo abs(F) dove F=(w^n)-z');
figure % figura 2
ezmesh(abs(F), [-1.2 1.2]); % DISEGNO DEL MODULO DI f IN 3D con ezmesh()
title('Funzione ezmesh( ) su Modulo abs(F) dove F=(w^n)-z');
% E' POSSIBILE DISEGNARE IN 3D LA FUNZIONE USANDO LE COORDINATE
% CARTESIANE
% - PARTE REALE real(F)
% - PARTE IMMAGINARIA imag(F)
% MA IN TAL CASO BISOGNA EFFETTUARE 2 DISEGNI
figure % figura 3
ezsurf(real(F), [-1.2 1.2]); % DISEGNO PARTE REALE DI f
title('Funzione ezsurf( ) su Parte Reale real(F) dove F=(w^n)-z');
figure % figure 4
ezsurf(imag(F), [-1.2 1.2]); % DISEGNO PARTE IMMAGINARIA DI f
title('Funzione ezsurf( ) su Parte Immaginaria imag(F) dove F=(w^n)-z');
```

Esempio d’uso

```
Inserire un Numero Complesso z (nella forma a+ib) = 2+3i
Inserire Grado della Radice n = 3
```

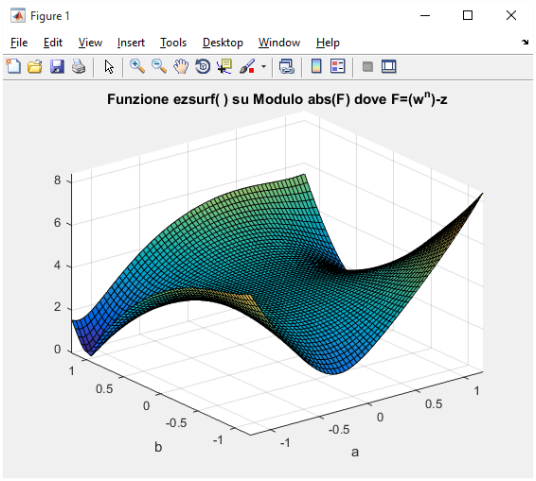


Figure 1
File Edit View Insert Tools Desktop Window Help
Funzione ezsurf( ) su Modulo abs(F) dove F=(w^n)-z
b
a

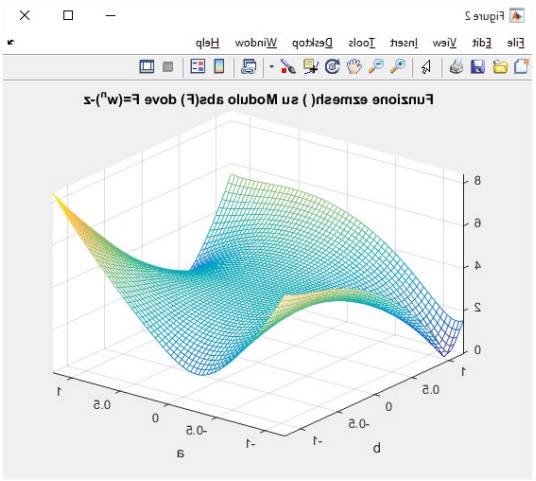


Figure 2
Funzione ezsurf( ) su Modulo abs(F) dove F=(w^n)-z

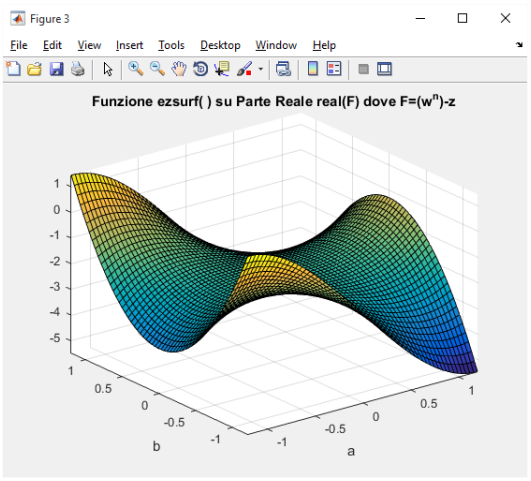


Figure 3
File Edit View Insert Tools Desktop Window Help
Funzione ezsurf( ) su Parte Reale real(F) dove F=(w^n)-z
b
a

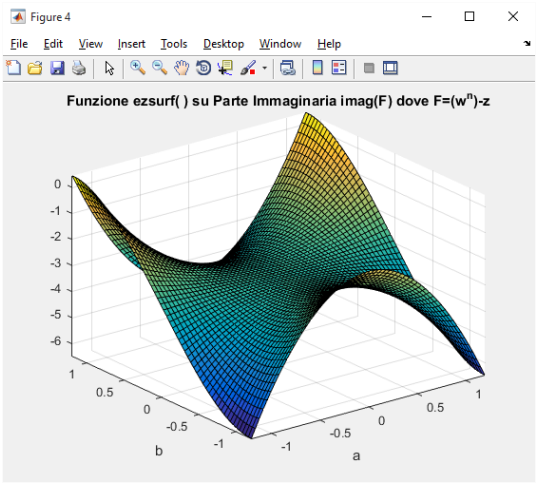


Figure 4
File Edit View Insert Tools Desktop Window Help
Funzione ezsurf( ) su Parte Immaginaria imag(F) dove F=(w^n)-z
b
a

ES. 5 – Liv. 1 – Primitive

Scrivere una *function* che, fissato un valore di n, restituisca tutte le radici n^{sime} dell'unità primitive.

Realizzare due algoritmi: il primo, mediante le coordinate polari, usa direttamente la proprietà matematica che le caratterizza ed il secondo invece ricorre alla funzione roots(). Confrontare i risultati.

Soluzione presentata

L'algoritmo si compone di un programma chiamante e di due funzioni.

La prima funzione, denominata ***f_proots()***, prende in input il grado n delle radici unitarie da calcolare e restituisce in output le radici n-esime primitive dell'unità. Il calcolo delle radici unitarie viene effettuato con la funzione *roots()*, che, essendo una function MATLAB, rammentiamo, opera nell'intervallo principale [-pi, pi]. Occorrerà pertanto riordinare le radici unitarie secondo il medesimo ordine adottato dal calcolo tramite formula matematica, in modo da poter poi restituire in output le radici primitive con lo stesso ordine della formula matematica.

La seconda funzione, denominata ***f_ppolari( )*** , prende in input il grado n delle radici unitarie da calcolare e restituisce in output le radici n-esime primitive dell'unità. Il calcolo delle radici unitarie viene effettuato con la regola matematica ….

$$w_k = \sqrt[n]{z} = [r, \varphi_k] = \begin{cases} r = \sqrt[n]{z} \\ \varphi_k = \dfrac{\theta}{n} \pm \dfrac{2k\pi}{n}, \quad k = 0,1, \dots n-1 \end{cases} .$$

che considera l'intervallo principale [0, 2pi].

```
%   Matematica Applicata e Computazionale
%
%   Primitive - Liv 1
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
% USO
%
% Lanciare da consolle il file
%
% >> primitive
%
% Inserire Grado di Radice = 5
%
% e vedere i calcoli che esegue


%%%%%%%
% MAIN
%%%%%%%

clc % Pulisci la schermmata MATLAB

n=input('Inserire Grado di Radice =');
primitive_roots=f_proots(n) % carica la funzione f_proots
hold on
primitive_polari=f_ppolari(n) % carica la funzione f_ppolari
```

```
%
%   Matematica Applicata e Computazionale
%
%   Primitive - Liv 1
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%

% %%%%%%%%%%%%%%%%%%%%
% Funzione f_proots.m
% %%%%%%%%%%%%%%%%%%%%

function [primitive_roots] = f_proots(n)
k=(0:n-1)';
r_unit=roots([1 zeros(1,n-1) -1])

% LE RADICI OTTENUTE CON roots() VANNO PRIMA ORDINATE CON sort E circshift
% PER DARE STESSO ORDINAMENTO IN [0,2pi] CHE HANNO LE
% RADICI UNITARIE OTTENUTE CON LA FORMULA CLASSICA
% E POI APPLICARE LO STESSO ALGORITMO
[r_sort,jj]=sort(angle(r_unit));

% TROVO L'INDICE f DI POSIZIONE DEL PRIMO ANGOLO MAGGIORE DI 0
% PER RUOTARE CON circshift() L'ARRAY DI UN NUMERO DI POSIZIONI CORRETTO
f=find(angle(r_unit(jj))>=0,1,'first');
r_unit=circshift(r_unit(jj),-(f-1)); % ROTAZIONE DELLE RADICI COMPLESSE
circshift()

% UNA RADICE UNITARIA E' PRIMITIVA SE IL SUO GRADO E IL GRADO n-esimo
MCD=gcd(n,k); % SONO TRA LORO PRIMI
```

```
% POSIZIONI DELLE RADICI PRIMITIVE NELL'ARRAY r_unit RIORDINATO
k_prim=k(MCD==1);
k_prim=k_prim+1; % esclude 1 dalle radici primitive

% seleziona in base agli indici di posizione le radici primitive
primitive_roots=r_unit(k_prim);
h=compass(primitive_roots,'r');
set(h,'LineWidth',4);

end % end function
```

```
%   Matematica Applicata e Computazionale
%
%   Primitive - Liv 1
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Funzione f_ppolari.m
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ primitive_polari ] = f_ppolari(n)
k=(0:n-1)';

% VETTORE UNITA'
R=1^(1/n); %MODULO DELL'UNITA'
theta=0; %ARGOMENTO DELL'UNITA'

% argomento delle radici unitarie(nella FORMULA MATEMATICA)
phi=2*k*pi/n;
% radici unitarie
w_unit=R*exp(1i*phi); %FORMULA ESPONENZIALE POLARE
% UNA RADICE UNITARIA E' PRIMITIVA SE IL SUO GRADO E IL GRADO n-esimo
% SONO TRA LORO PRIMI
MCD=gcd(n,k);
k_prim=k(MCD==1);
k_prim=k_prim+1; % esclude 1 dalle radici primitive

% POSIZIONI DELLE RADICI PRIMITIVE
% seleziona in base agli indici di posizione le radici primitive
```

```
primitive_polari=w_unit(k_prim);

% GRAFICO
h=compass(primitive_polari,'c');
set(h,'LineWidth',2);

end % end function
```

Esempio d'uso

```
Inserire Grado di Radice =5

r_unit =

  -0.8090 + 0.5878i
  -0.8090 - 0.5878i
   0.3090 + 0.9511i
   0.3090 - 0.9511i
   1.0000 + 0.0000i


primitive_roots =

   0.3090 + 0.9511i
  -0.8090 + 0.5878i
  -0.8090 - 0.5878i
   0.3090 - 0.9511i


primitive_polari =

   0.3090 + 0.9511i
  -0.8090 + 0.5878i
  -0.8090 - 0.5878i
   0.3090 - 0.9511i
```

Graficamente segue

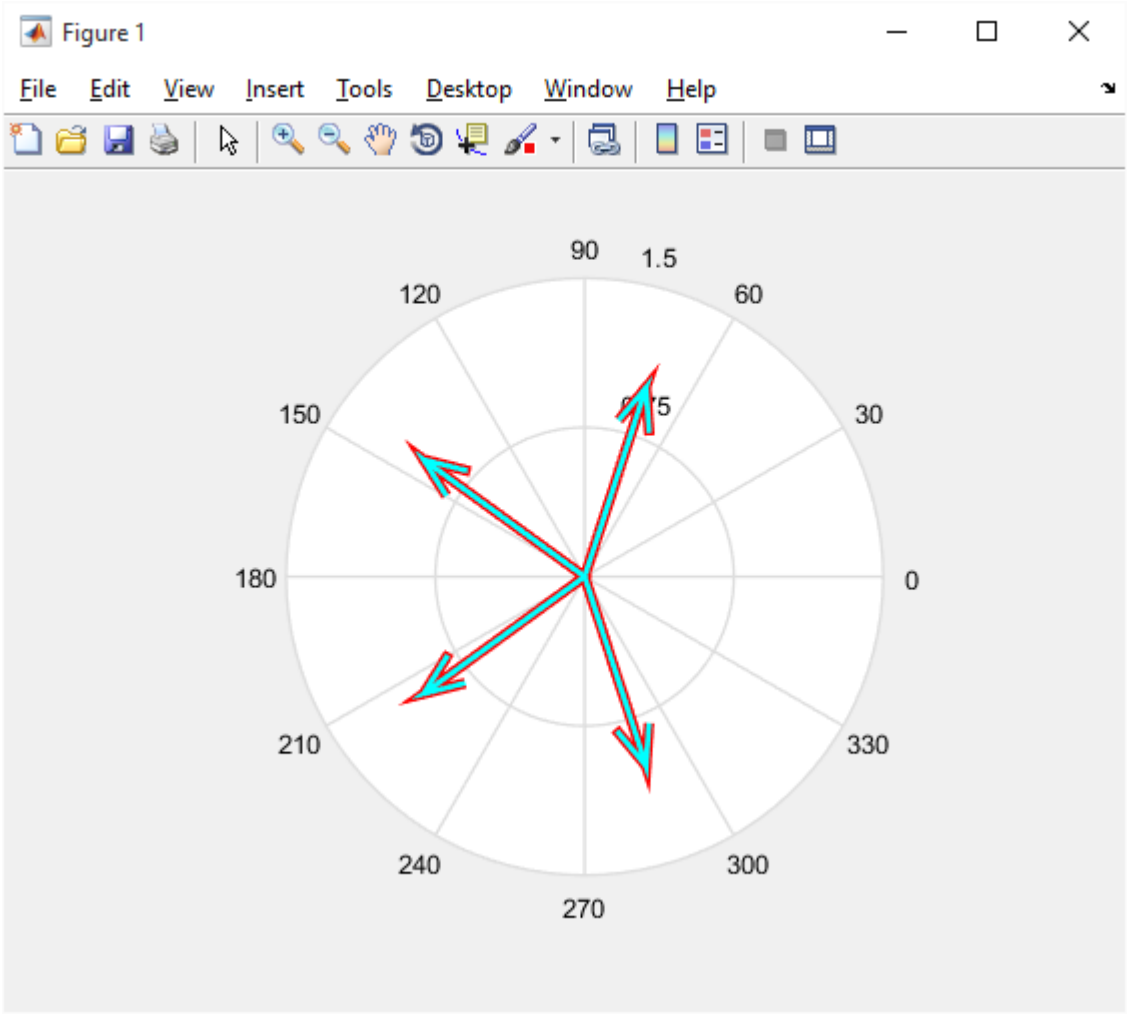


Figure 1
File Edit View Insert Tools Desktop Window Help
90
1.5
120
60
150
30
180
0
210
330
240
300
270

ES. 6 – Primitive

Liv- 1 – Calcolare tutte le radici n^{sime} di un numero complesso tramite le potenze successive di una qualsiasi radice n^{sima} dell'unità primitiva scelta a caso o dall'utente.

Liv- 2 – Visualizzare graficamente l'ordine in cui sono generate le radici.

Soluzione Proposta

L'algoritmo, che risponde ad entrambi i quesiti della traccia, è composto da un programma chiamante, che riceve in input il numero complesso z ed il grado n delle radici da calcolare, e poi viene richiamata la function **esercizio_06l1( )** che risponde ai quesiti del primo esercizio.

Modificando il programma chiamante **potradcomplex.m** (e come commentato nel sorgente) si può decidere di chiamare il primo ed il secondo esercizio (**esercizio_06l1( ) ed esercizio_06l2( ))**

Come si può osservare consultando la sezione **Esempi d'uso**, dopo aver digitato i parametri di input, la funzione calcola le radici primitive dell'unità mediante coordinate polari.

Successivamente, viene scelta a caso una di tali radici che verrà poi moltiplicata per una radice particolare di z in modo da ottenere le radici n^{sime} del numero complesso z, generate mediante potenze successive della radice n^{sima} dell'unità primitiva scelta precedentemente a caso.

A conferma della bontà del risultato ottenuto, vengono poi mostrate in output le radici n-sime di z calcolate con `roots( )` . Naturalmente l'ordine è diverso, perché nel 1 caso è stata usata la formula matematica che lavora nell'intervallo [0 2pi] mentre rammentiamo che roots usa gli argomenti principali.

Vengono poi disegnate le radici di z e, con un secondo grafico, viene poi mostrato l'ordine con cui tali radici sono state generate

```
%
%   Matematica Applicata e Computazionale
%
%   Calcola le potenze n-sime di Radici Complesse - Liv 1
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
% USO
%
% Lanciare da consolle il file
%
% >> potradcomplex
%
% Inserire il numero complesso z = 4-3i
%
% Inserire Grado n delle Radici da calcolare = 3
%
% e vedere i calcoli che esegue

%%%%%%%
% MAIN
%%%%%%%
clc % Pulisci la schermmata MATLAB

z=input('Inserire il numero complesso z = ');
n=input('Inserire Grado n delle Radici da calcolare = ');
% CALCOLA RADICI ENNESIME COME POTENZE SUCCESSIVE DI UNA PRIMITIVA A SCELTA
radici = esercizio6l1 (z, n);
% Basta cambiare su ^ il nome della funzione da esercizio6l1 a esercizio6l2
% per risolvere anche il secondo esercizio
```

```
%
%   Matematica Applicata e Computazionale
%
%   Calcola le potenze n-sime di Radici Complesse - Liv 1
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%

%%%%%%%%%%%%%%%%%%%%%%%%%%
% function esercizio6l1.m
%%%%%%%%%%%%%%%%%%%%%%%%%%
function radici = esercizio6l1 (z, n)

p = []; % Array che conterrà gli indici delle radici dell'unità primitive

% Le radici n-sime dell'unità sono primitive se il Massimo Comun Divisore
% tra n e k (con k=0,1,...,n-1) è uguale a 1. In realtà, per k=0, otteniamo
% come risultato 1 che, pur essendo radice n-sima dell'unità, non è
% primitiva.
% La prima radice n-sima dell'unità primitiva si ottiene con k=1, cioè i.
% Il ciclo for seguente salverà all'interno dell'array p gli indici k per
% i quali si ottengono solo radici n-sime dell'unità primitive.

for k = (1:n-1)
      if (gcd(n,k) == 1) % Se MCD tra n e k è 1...
            p = [p k];   % ...salva l'indice k nell'array p
      end % end if
end % end for

% Adesso, calcoliamo le radici n-sime dell'unità primitive mediante
coordinate polari
theta = (2*pi*p')/(n);
```

```
risultato_polari = exp(1i*theta);
disp ('Radici primitive dell''unità mediante coordinate polari:');
disp (risultato_polari);

% Generiamo un numero casuale nell'intervallo [1; p] che verrà, poi, usato
% come indice per la selezione della radice n-sima dell'unità primitiva
% corrispondente nel vettore risultato_polari.
j = 1+round(rand*(length(p)-1));
disp ('Radice n-sima dell unità primitiva scelta in modo casuale:');
disp (risultato_polari(j));

% Per calcolare tutte le radici del numero complesso z basta moltiplicare
% una sua radice particolare per una delle radici n-sime dell'unità
% primitive elevata a potenze successive (k=0,1,2,...,n-1)
k = (0:n-1)';
radici = z^(1/n)*(risultato_polari(j).^k);
disp ('Radice n-sima particolare del numero complesso z:');
eval ('z^(1/n)');
disp ('Radici n-sime del numero complesso z, generate mediante potenze
successive di una radice n-sima dell unità primitiva scelta a caso:');
disp (radici);
% Costruzione del grafico per visualizzare le radici n-sime del numero
% complesso z.
figure;
h = compass(radici);
set(h, 'LineWidth', 3);
title ('Radici n-sime del numero complesso z');

% Come conferma delle operazioni, calcoliamo le radici n-sime del numero
% complesso z mediante la function 'roots'.
radici_roots = roots([1 zeros(1,n-1) -z]);
disp ('Radici n-sime di un numero complesso z, generate mediante
roots():');
disp (radici_roots);
```

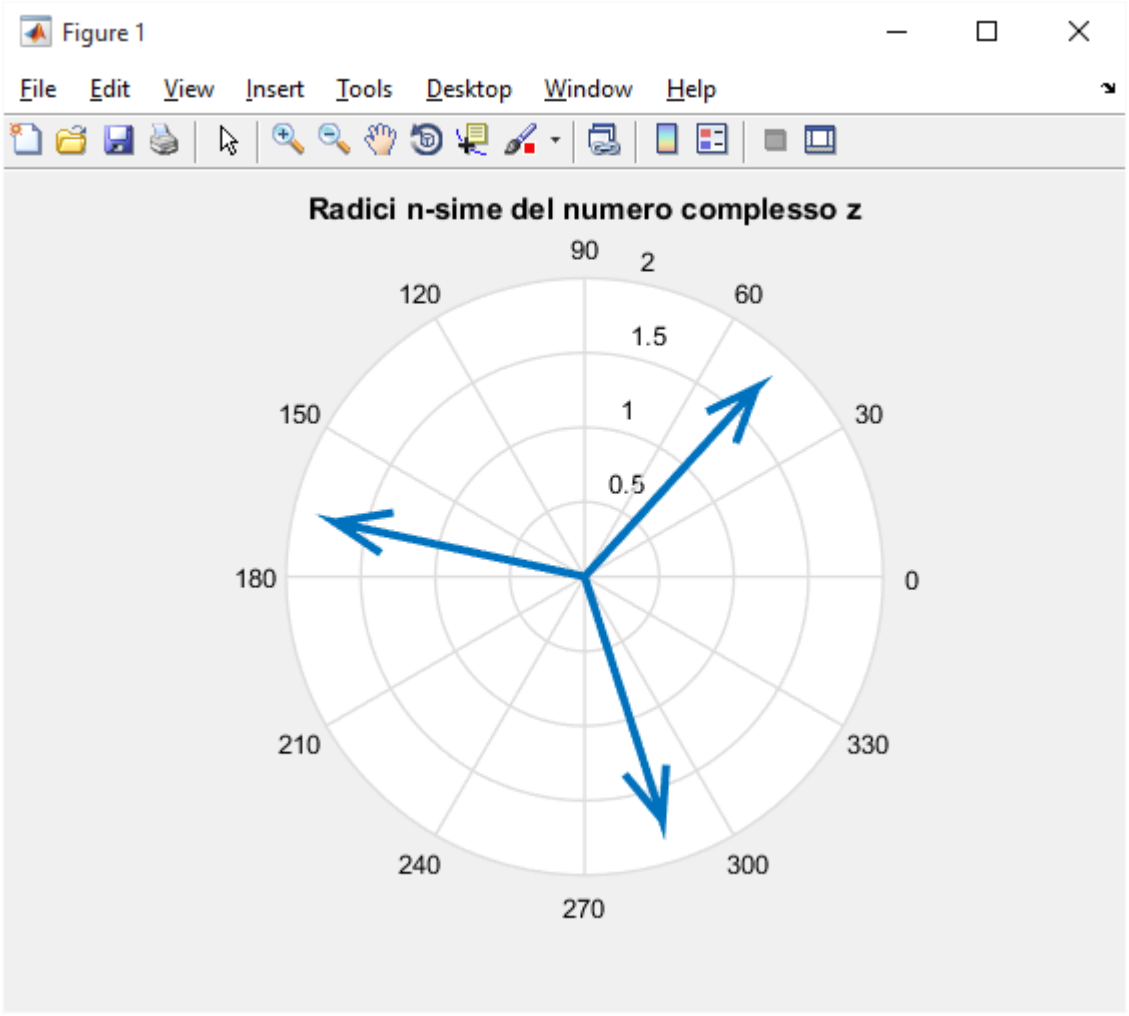
```
%   Matematica Applicata e Computazionale
%
%   Calcola le potenze n-sime di Radici Complesse - Liv 2
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%

%%%%%%%%%%%%%%%%%%%%%%%%%%
% function esercizio6l2.m
%%%%%%%%%%%%%%%%%%%%%%%%%%
function radici = esercizio6l2 (z, n)

p = []; % Array che conterrà gli indici delle radici dell'unità primitive

% Le radici n-sime dell'unità sono primitive se il Massimo Comun Divisore
% tra n e k (con k=0,1,...,n-1) è uguale a 1. In realtà, per k=0, otteniamo
% come risultato 1 che, pur essendo radice n-sima dell'unità, non è
primitiva.
% La prima radice n-sima dell'unità primitiva si ottiene con k=1, cioè i.
% Il ciclo for seguente salverà all'interno dell'array p gli indici k per
% i quali si ottengono solo radici n-sime dell'unità primitive.

for k = (1:n-1)
      if (gcd(n,k) == 1) % Se MCD tra n e k è 1...
            p = [p k]; % ...salva l'indice k nell'array p
      end
end

% Adesso, calcoliamo le radici n-sime dell'unità primitive mediante
% coordinate polari
theta = (2*pi*p')/(n);
risultato_polari = exp(1i*theta);
```

```
disp ('Radici primitive dell''unità mediante coordinate polari:');
disp (risultato_polari);

% Generiamo un numero casuale nell'intervallo [1; p] che verrà, poi, usato
% come indice per la selezione della radice n-sima dell'unità primitiva
% corrispondente nel vettore risultato_polari.
j = 1+round(rand*(length(p)-1));
disp ('Radice n-sima dell unità primitiva scelta in modo casuale:');
disp (risultato_polari(j));

% Per calcolare tutte le radici del numero complesso z basta moltiplicare
% una sua radice particolare per una delle radici n-sime dell'unità
% primitive elevata a potenze successive (k=0,1,2,...,n-1)
k = (0:n-1)';
radici = z^(1/n)*(risultato_polari(j).^k);
disp ('Radice n-sima particolare del numero complesso z:');
eval ('z^(1/n)');
disp ('Radici n-sime del numero complesso z, generate mediante potenze
successive di una radice n-sima dell unità primitiva scelta a caso:');
disp (radici);
% Costruzione del grafico per visualizzare le radici n-sime del numero
% complesso z.
figure;
h = compass(radici);
set(h, 'LineWidth', 3);
title ('Radici n-sime del numero complesso z');

% Come conferma delle operazioni, calcoliamo le radici n-sime del numero
% complesso z mediante la function 'roots'.
radici_roots = roots([1 zeros(1,n-1) -z]);
disp ('Radici n-sime di un numero complesso z, generate mediante
roots():');
disp (radici_roots);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Quesito di liv 2: ordine in cui sono generate le radici
% Costruzione del grafico per visualizzare l'ordine in cui sono generate
% le radici.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure;
for k = (1:n)
      h1 = compass (radici(k), 'r');
```

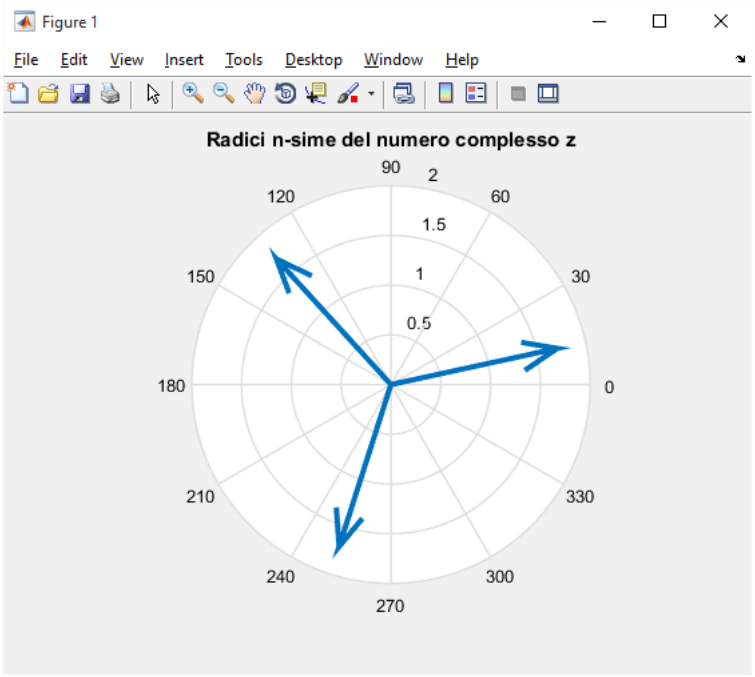
```
      axis equal;
      hold on;
      set(h1, 'LineWidth', 3);
      plot (radici,'k', 'LineWidth', 1.5);
      title ('Ordine delle radici n-sime del numero complesso z');
      pause(2);
end % end for
```

Esempio d'uso – Livello 1

```
Inserire il numero complesso z = -4+3i
Inserire Grado n delle Radici da calcolare = 3
Radici primitive dell'unitÃ  mediante coordinate polari:
  -0.5000 + 0.8660i
  -0.5000 - 0.8660i

Radice n-sima dell unitÃ  primitiva scelta in modo casuale:
  -0.5000 - 0.8660i

Radice n-sima particolare del numero complesso z:

ans =

   1.1506 + 1.2650i

Radici n-sime del numero complesso z, generate mediante potenze successive
di una radice n-sima dell unità primitiva scelta a caso:
   1.1506 + 1.2650i
   0.5202 - 1.6289i
  -1.6708 + 0.3640i

Radici n-sime di un numero complesso z, generate mediante roots():
  -1.6708 + 0.3640i
   0.5202 - 1.6289i
   1.1506 + 1.2650i
```



Esempio d’uso – Livello 2



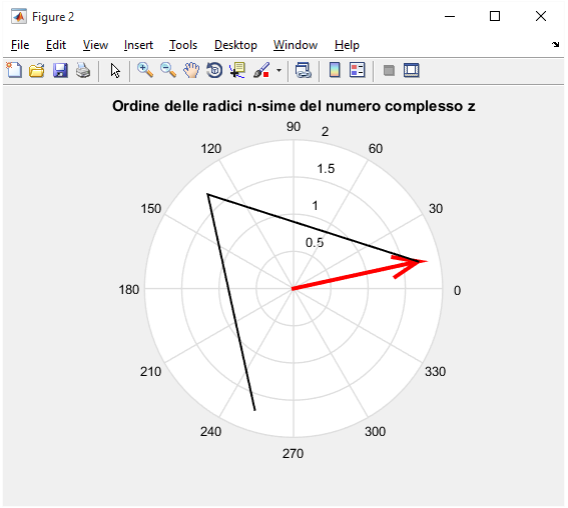


Figure 2
File Edit View Insert Tools Desktop Window Help
Ordine delle radici n-sime del numero complesso z
90
2
120
60
1.5
150
1
30
0.5
180
0
210
330
240
300
270

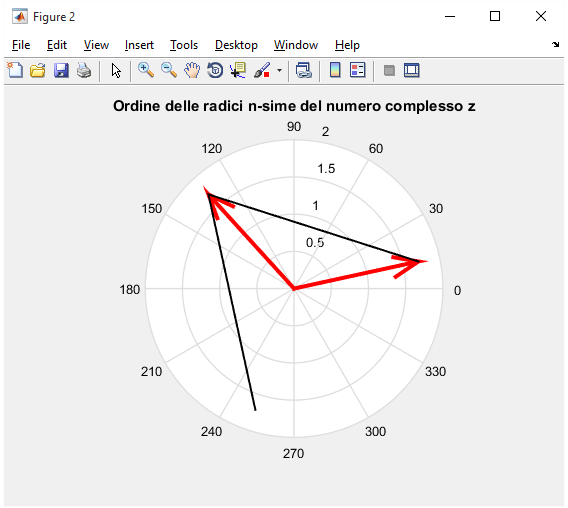


Figure 2
File Edit View Insert Tools Desktop Window Help
Ordine delle radici n-sime del numero complesso z
90
2
120
60
1.5
150
1
30
0.5
180
0
210
330
240
300
270

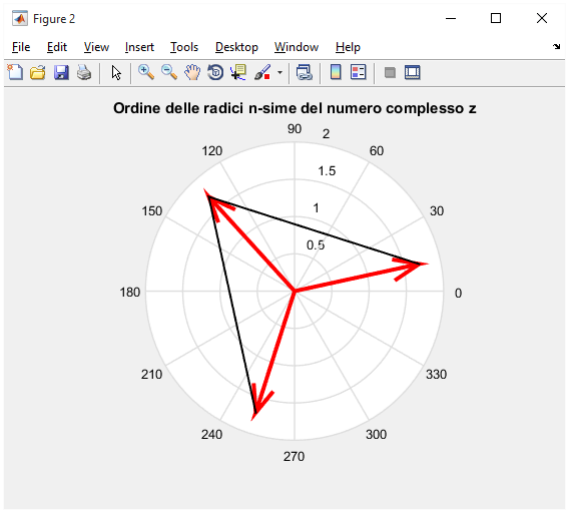


Figure 2
File Edit View Insert Tools Desktop Window Help
Ordine delle radici n-sime del numero complesso z
90
2
120
60
1.5
150
1
30
0.5
180
0
210
330
240
300
270

Funzioni nel campo complesso

A – Funzioni complesse di variabili reali

ES. 7 – Liv. 1 – Funzioni complesse di variabili reali

Visualizzare mediante il MATLAB Symbolic Math Toolbox alcune curve del piano (…, ellisse, arco di iperbole [cercare sul web le relative equazioni parametriche]) descritte mediante equazioni parametriche. Nel caso di curve regolari, visualizzare anche la tangente in un suo punto scelto dall'utente (oppure scelto a caso).

```
%
%   Matematica Applicata e Computazionale
%
%   Funzioni complesse di variabili reali - Liv 1
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
% USO
%
% Lanciare da consolle il file
%
% >> funcomplexreal
%
% Inserire il numero complesso z = 4-3i
%
% Inserire Grado n delle Radici da calcolare = 3
%
```

```
% e vedere i calcoli che esegue

clc % Pulisci la schermmata MATLAB

% Scelta della curva del piano
scelta = menu ('Scegli la curva', 'Circonferenza', 'Ellisse', 'Parabola');

% Usiamo lo switch per effettuare le scelte
switch scelta
      %%%%%%%%%%%%%%%%%%%%%%%
      % Case 1 Circonferenza
      %%%%%%%%%%%%%%%%%%%%%%%
      case 1 % Circonferenza
            circonferenza(); % carica la funzione circonferenza

      %%%%%%%%%%%%%%%%%%
      % Case 2 Ellisse
      %%%%%%%%%%%%%%%%%%
      case 2 % Ellisse
            ellisse (); % Carica la funzione ellisse


      %%%%%%%%%%%%%%%%%%%
      % Case 3 Parabola
      %%%%%%%%%%%%%%%%%%%
      case 3 % Parabola
            parabola ();

end % End Switch
```

```
%
%   Matematica Applicata e Computazionale
%
%   Funzioni complesse di variabili reali - Liv 1
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%

%%%%%%%%%%%%%%%%%%%%%%%%%
% Funzione Circonferenza
%%%%%%%%%%%%%%%%%%%%%%%%%

function circonferenza

            clc % Pulisci la schermmata MATLAB

            disp('Hai scelto la circonferenza');
            syms t real; % Dichiarazione del parametro
            r = input('Inserisci il raggio:');

            disp('Inserisci le coordinate del centro:');
            x_0 = input('Ascissa:');
            y_0 = input('Ordinata:');

            disp('Le equazioni parametriche della Circonferenza sono:');
            x = x_0+r*cos(t);
            y = y_0+r*sin(t);

            pretty(x); % Stampa le equazioni parametriche (in simbolico)
                       % in "forma matematica"
            pretty(y);
```

```
            zt = x+i*y; % Equazione parametrica della circonferenza
            h = ezplot(real(zt),imag(zt)); % Disegna la circonferenza
            axis([x_0-r-1 x_0+r+1 y_0-r-1 y_0+r+1]);
            set(h,'LineWidth',2);
            title ('Circonferenza')
            hold on;

            risposta = input ('Digita 1 per immettere manualmente il valore
del parametro t.\nDigita 2 per generare il parametro in modo random.\nFai
la tua scelta: ');
            if (risposta == 1)
                  t1 = input('Scegliere un valore per il parametro compreso
tra 0 e 360:');
            else
                  t1=round(rand*(360));
                  disp ('Il parametro selezionato casualmente è ');
                  disp (t1);
            end
            t0 = (t1*pi)/180; % Conversione del parametro immesso in
                              % radianti
            z_t0 = subs(zt,'t',t0); % Sostituiamo il parametro
                                    % nell'equazione della circonferenza

            % Visualizziamo il punto fissato sulla Circonferenza
            plot(real(z_t0),imag(z_t0),'or')

            % Calcoliamo la derivata della funzione
            z1t = diff(zt);

            % Sostituiamo il parametro t0 nella derivata in modo da
            % ottenere la direzione della tangente in quel punto
            z1_t0 = subs(z1t,'t',t0);

            % Scriviamo l'equazione della retta tangente alla circonferenza
            % nel particolare punto
            zTang = z_t0+z1_t0*t;

            % Disegnamo la retta tangente
            h1 = ezplot(real(zTang),imag(zTang),[-2 2]);
            axis equal;
            set(h1,'LineWidth',2,'Color','r')
            title('Circonferenza');
```

```
%
%   Matematica Applicata e Computazionale
%
%   Funzioni complesse di variabili reali - Liv 1
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%

%%%%%%%%%%%%%%%%%%%
% Funzione Ellisse
%%%%%%%%%%%%%%%%%%%

function ellisse

clc % Pulisci la schermmata MATLAB

disp('Hai scelto l''ellisse');

syms t real; % Dichiarazione del parametro

disp('Inserire le misure dei semiassi:');
a = input('Semiasse maggiore:');
b = input('Semiasse minore:');

disp('Le equazioni parametriche dell Ellisse sono:');
x = a*cos(t);
y = b*sin(t);

pretty(x); % Stampa le equazioni parametriche (in simbolico) in "forma
           % matematica"
pretty(y);
```

```
zt = x+i*y; % Equazione parametrica dell'ellisse

h = ezplot(real(zt),imag(zt)); % Disegna l'ellisse

axis ([-a-2 a+2 -b-2 b+2]);
set(h,'LineWidth',2);
title ('Ellisse');
hold on;
risposta = input ('Digita 1 per immettere manualmente il valore del parametro t.\nDigita 2 per generare il parametro in modo random.\nFai la tua scelta: ');

if (risposta == 1)
      t1 = input('Scegliere un valore per il parametro compreso tra 0 e 360:');
else
      t1=round(rand*(360));
      disp ('Il numero scelto è ');
      disp (t1);
end

t0 = (t1*pi)/180; % Conversione del parametro immesso in radianti

z_t0 = subs(zt,'t',t0); % Sostituiamo il parametro nell'equazione
                        % dell'ellisse

% Visualizziamo il punto fissato sull'ellisse
plot(real(z_t0),imag(z_t0),'or')

% Calcoliamo la derivata della funzione
z1t = diff(zt);

% Sostituiamo il parametro t0 nella derivata in modo da ottenere la
% direzione della tangente in quel punto
z1_t0 = subs(z1t,'t',t0);

% Scriviamo l'equazione della retta tangente all'ellisse nel particolare
% punto
zTang = z_t0+z1_t0*t;
```

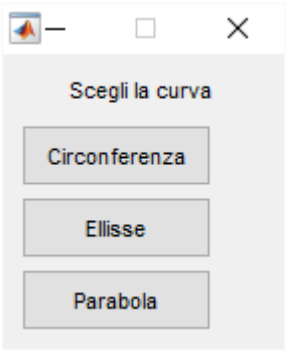
```
% Disegnamo la retta tangente
h1 = ezplot(real(zTang),imag(zTang),[-2 2]);
axis equal;
set(h1,'LineWidth',2,'Color','r')
title('Ellisse');

end % end Function
```

```
%
%   Matematica Applicata e Computazionale
%
%   Funzioni complesse di variabili reali - Liv 1
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%

%%%%%%%%%%%%%%%%%%%
% Funzione Parabola
%%%%%%%%%%%%%%%%%%%

function parabola

clc % Pulisci la schermmata MATLAB

disp('Hai scelto la parabola');
syms t real; % Dichiarazione del parametro
disp('Le equazioni parametriche della parabola sono:');
x = t;
y = t^2;

pretty(x); % Stampa le equazioni parametriche (in simbolico) in "forma
           % matematica"
pretty(y);

zt = x+i*y; % Equazione parametrica della parabola
h = ezplot(real(zt),imag(zt),[-2 2]); % Disegna la parabola nell'intervallo
[-2, 2]
axis equal;
set(h,'LineWidth',2);
```

```
title ('Parabola')
hold on
risposta = input ('Digita 1 per immettere manualmente il valore del
parametro t.\nDigita 2 per generare il parametro in modo random.\nFai la
tua scelta: ');

if (risposta == 1)
      t0 = input('Scegliere un valore per il parametro compreso tra - 2 e
2:');
else
      t0 = unifrnd(-2,2); % unifrnd restituisce un numero casuale scelto
% dalla distribuzione continua uniforme nell'intervallo specificato, che
% è l'intervallo in cui viene disegnata la parabola
      disp ('Il numero scelto è ');
      disp (t0);
end

z_t0 = subs(zt,'t',t0); % Sostituiamo il parametro nell'equazione della
% parabola,per vedere t0 a quale punto della curva corrisponde

% Visualizziamo il punto fissato sulla parabola
plot(real(z_t0),imag(z_t0),'or')

% Calcoliamo la derivata della funzione,per stabilire quale direzione avrà
la retta tangente, che passa per z_t0
z1t = diff(zt);
% Sostituiamo il parametro t0 nella derivata in modo da ottenere la
% direzione della tangente in quel punto, ovvero il coeff angolare
z1_t0 = subs(z1t,'t',t0);

% Scriviamo l'equazione della retta tangente alla parabola nel particolare
punto
zTang = z_t0+z1_t0*t;

% Disegnamo la retta tangente
h1 = ezplot(real(zTang),imag(zTang),[-2 2]);
axis equal;
set(h1,'LineWidth',2,'Color','r')
title('Parabola');

end % end Function
```

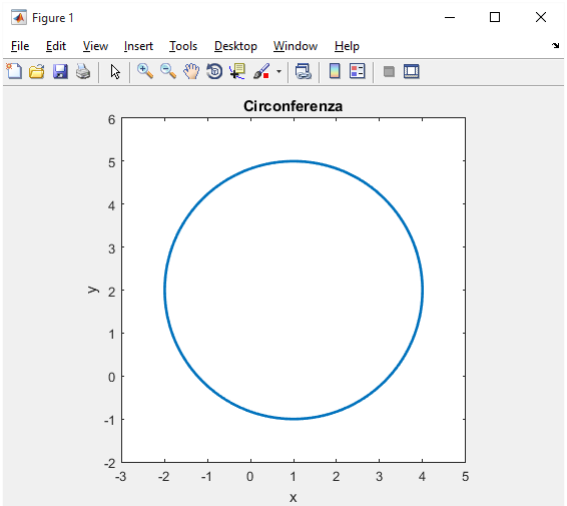
Esempi d'uso

Fatto partire il programma dobbiamo scegliere dal ToolBox la curva da usare

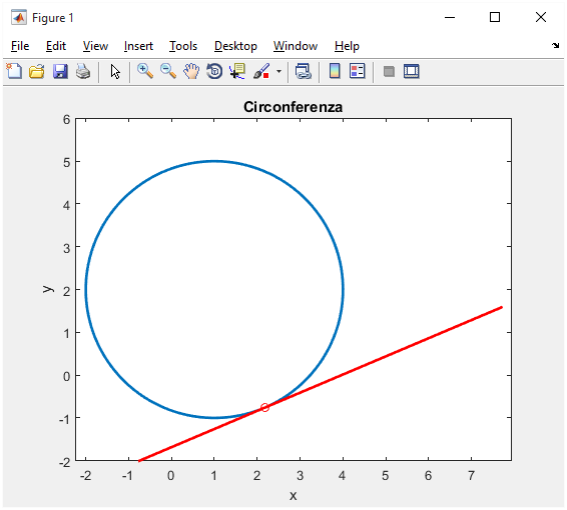


Le vedremo una ad una, partiamo dalla Circonferenza usando i valori random

```
Hai scelto la circonferenza

Inserisci le coordinate del centro:
Ascissa:1
Ordinata:2
Le equazioni parametriche della Circonferenza sono:
3 cos(t) + 1

3 sin(t) + 2

Digita 1 per immettere manualmente il valore del parametro t.
Digita 2 per generare il parametro in modo random.
Fai la tua scelta:2
```

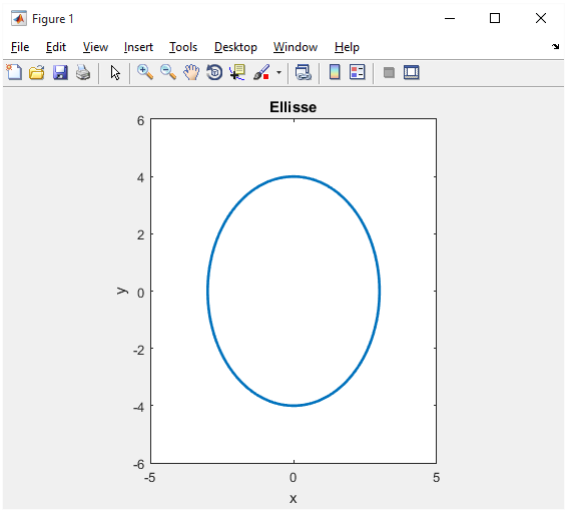


```
Il parametro selezionato casualmente è
   293
```

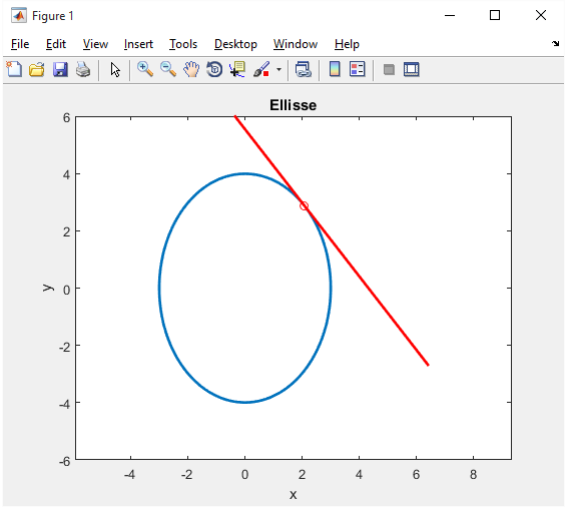


Ellisse, facciamo ripartire il programma

```
Hai scelto l'ellisse

Semiasse maggiore:3
Semiasse minore:4
Le equazioni parametriche dell Ellisse sono:
3 cos(t)

4 sin(t)

Digita 1 per immettere manualmente il valore del parametro t.
Digita 2 per generare il parametro in modo random.
Fai la tua scelta:
```

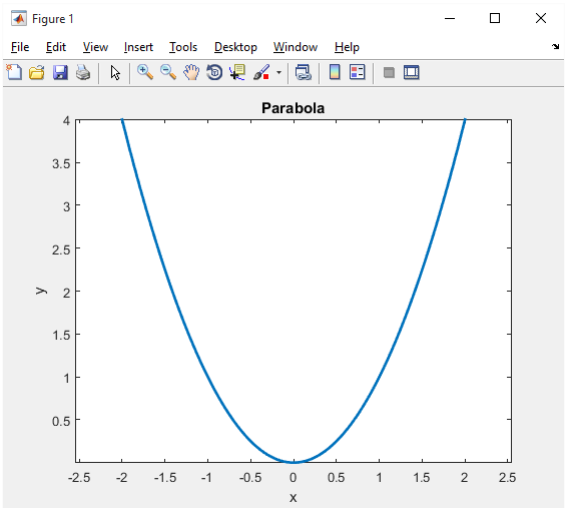


```
Digita 1 per immettere manualmente il valore del parametro t.
Digita 2 per generare il parametro in modo random.
Fai la tua scelta: 2
Il numero scelto è
    46
```

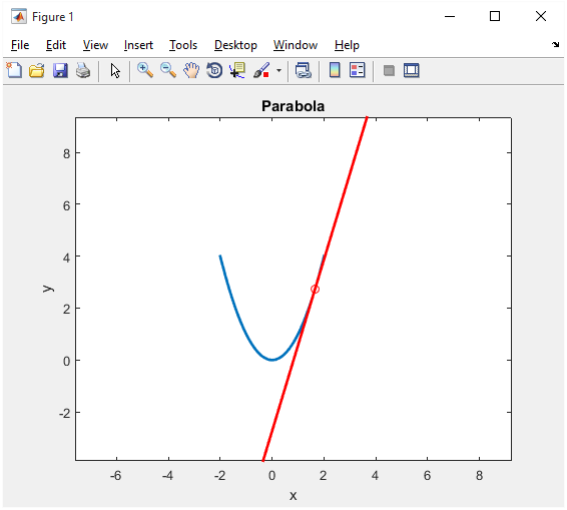


Parabola, facciamo ripartire il programma

```
Le equazioni parametriche della parabola sono:
t

 2
t

Digita 1 per immettere manualmente il valore del parametro t.
Digita 2 per generare il parametro in modo random.
Fai la tua scelta:
```



```
Digita 1 per immettere manualmente il valore del parametro t.
Digita 2 per generare il parametro in modo random.
Fai la tua scelta: 2
Il numero scelto è
    1.6535
```



ES. 8 – Liv. 1 – Funzioni complesse di variabili reali

Ripetere in ambiente numerico l'esercizio precedente, sapendo che ora non si ha l'espressione della funzione ma solo n punti campioni.

Soluzione Proposta

L'algoritmo che segue è molto simile a quello del precedente esercizio.
La differenza sta nel fatto che, affrontando il problema in ambito numerico, come punto di partenza non c'è più l'equazione parametrica della curva, ma partiamo da un insieme di valori detti CAMPIONI DELLA CURVA.
Ciò significa che, nel calcolo della derivata, partendo dai campioni, dovremo effettuarne una APPROSSIMAZIONE, ricorrendo ad un RAPPORTO INCRENMENTALE, con una istruzione MATLAB del tipo …

```
% L'istruzione calcola i rapporti incrementali relativi ai punti
% generati. diff(Zk) rappresenta la misura del segmento tra 2 campioni
% successivi, mentre diff(Tk) l'incremento
z1t = diff(Zk)./diff(Tk);
```

Dunque, la differenza sostanziale tra numerico e simbolico sta nel fatto che in ambiente numerico le derivate sono approssimate, ragion per cui il loro valore dipenderà dai campioni iniziali.

Sostanzialmente il Main rimane uguale al precedente, ma è riproposto integralmente tutto il sorgente del programma

```
%
%   Matematica Applicata e Computazionale
%
%   Funzioni complesse di variabili reali - Liv 1
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
% USO
%
% Lanciare da consolle il file
%
% >> funcomplexreal
%
% Inserire il numero complesso z = 4-3i
%
% Inserire Grado n delle Radici da calcolare = 3
%
% e vedere i calcoli che esegue

clc % Pulisci la schermmata MATLAB

% Scelta della curva del piano
scelta = menu ('Scegli la curva', 'Circonferenza', 'Ellisse', 'Parabola');

% Usiamo lo switch per effettuare le scelte
switch scelta
      %%%%%%%%%%%%%%%%%%%%%%%
      % Case 1 Circonferenza
      %%%%%%%%%%%%%%%%%%%%%%%
      case 1 % Circonferenza
            circonferenza(); % carica la funzione circonferenza
```

```
      %%%%%%%%%%%%%%%%%
      % Case 2 Ellisse
      %%%%%%%%%%%%%%%%%
      case 2 % Ellisse
            ellisse (); % Carica la funzione ellisse


      %%%%%%%%%%%%%%%%%%
      % Case 3 Parabola
      %%%%%%%%%%%%%%%%%%
      case 3 % Parabola
            parabola ();

end % end Switch
```

```
%   Matematica Applicata e Computazionale
%
%   Funzioni complesse di variabili reali - Liv 1
%   per punti campione
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%

%%%%%%%%%%%%%%%%%%%%%%%%%
% Funzione Circonferenza
%%%%%%%%%%%%%%%%%%%%%%%%%

function circonferenza

clc % Pulisci la schermmata MATLAB

disp('Hai scelto la circonferenza');

% Scelta del numero di campioni n
n = input('Inserire il numero di campioni:');

% Determinazione di n campioni equispaziati tra 0 e 2pi greco
Tk = linspace(0,2*pi,n);
r = input('Inserisci il raggio:');
disp('Inserisci le coordinate del centro:');
x_0 = input('Ascissa:');
y_0 = input('Ordinata:');

% Centro della circonferenza come numero complesso
c = x_0+1i*y_0;
```

```
% Equazione parametrica della circonferenza in forma complessa
Zk = c+r*exp(1i*Tk);

% Costruzione della griglia di interpolazione
k = 1:length(Zk);
t = linspace(k(1),k(end),99); % Griglia equispaziata di 99 punti

% nei quali calcolare il valore dell'interpolante
% Interpoliamo la funzione Zk nei punti t sulla griglia k usando la function
% interp1.
% yy = interp1(X, FX, XX, TIPO), dove
% X è la griglia di interpolazione;
% FX sono i valori da interpolare;
% XX sono i punti sui quali si vuole calcolare l'interpolante
% TIPO è il tipo di interpolazione richiesta:
% - 'nearest' : interpolazione con polinomio costante a tratti
% - 'linear' : (default) polinomio lineare a tratti
% - 'cubic' : polinomio cubico a tratti (con derivate continue)
% - 'spline' : interpolazione con spline cubica
Zt = interp1(k,Zk,t,'spline');

plot(Zt); % Disegna la circonferenza
hold on;
axis equal;
h = plot (Zk,'*r'); % Disegna i campioni
axis([x_0-r-1 x_0+r+1 y_0-r-1 y_0+r+1]);
set(h,'LineWidth',2);
title ('Circonferenza')
risposta = input ('Digita 1 per scegliere manualmente il campione
t.\nDigita 2 per scegliere il campione in modo random.\nFai la tua scelta:
');
if (risposta == 1)
      j = input('Scegliere l''indice del campione t compreso tra 1 \ne il
massimo numero di campioni immesso n:');
else
      j = 1+round(rand*(n-1)); % Genera un intero casuale compreso
                               % tra 1 e n
      disp ('Il campione scelto ha indice');
      disp(j);
end

z_t0 = Zk(j); % Salva in z_t0 il j-simo campione della circonferenza
```

```
% Visualizziamo il punto fissato sulla circonferenza
plot(real(z_t0),imag(z_t0),'sk')

% L'istruzione calcola i rapporti incrementali relativi ai punti
% generati. diff(Zk) rappresenta la misura del segmento tra 2 campioni
% successivi, mentre diff(Tk) l'incremento
z1t = diff(Zk)./diff(Tk);

% Rapporto incrementale relativo al campione considerato.
% Esso costituirà il coefficiente angolare della retta tangente
z1_t0 = z1t(j);

% Scriviamo l'equazione della retta tangente alla circonferenza nel
% particolare punto
zTang = z_t0+z1_t0*[-1 2];

% [-1 2]Sostituisce t. Dato che per disegnare una retta occorrono
% due punti, è come se avessimo assegnato due valori al parametro t
% Disegnamo la retta tangente
h1 = plot(zTang);
axis equal;
set(h1,'LineWidth',2,'Color','k')
title('Circonferenza');

end % end function
```

```
%   Matematica Applicata e Computazionale
%
%   Funzioni complesse di variabili reali - Liv 1
%   per punti campione
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%


%%%%%%%%%%%%%%%%%%%
% Funzione Ellisse
%%%%%%%%%%%%%%%%%%%

function ellisse

clc % Pulisci la schermmata MATLAB

disp('Hai scelto l''ellisse');
% Scelta del numero di campioni n
n = input('Inserire il numero di campioni:');

% Determinazione di n campioni equispaziati tra -pi greco e pi greco
Tk = linspace(-pi,pi,n);
disp('Inserire le misure dei semiassi:');
a = input('Semiasse maggiore:');
b = input('Semiasse minore:');

% Equazione parametrica dell'ellisse in forma complessa
Zk = a*((exp(1i*Tk)+exp(-1i*Tk))/2)+1i*b*((exp(1i*Tk)-exp(-1i*Tk))/2*1i);
k = 1:length(Zk); % Costruzione della griglia di interpolazione
t=linspace(k(1),k(end),99); % Griglia equispaziata di 99 punti nei
```

```
% quali calcolare il valore dell'interpolante
% Interpoliamo la funzione Zk nei punti t sulla griglia k usando la function
% interp1.
% yy = interp1(X, FX, XX, TIPO), dove
% X è la griglia di interpolazione;
% FX sono i valori da interpolare;
% XX sono i punti sui quali si vuole calcolare l'interpolante
% TIPO è il tipo di interpolazione richiesta:
% - 'nearest' : interpolazione con polinomio costante a tratti
% - 'linear' : (default) polinomio lineare a tratti
% - 'cubic' : polinomio cubico a tratti (con derivate continue)
% - 'spline' : interpolazione con spline cubica
Zt = interp1(k,Zk,t,'spline');

plot(Zt); % Disegna l'ellisse
hold on;
axis equal

h = plot (Zk,'*r'); % Disegna i campioni
axis([-a-2 a+2 -b-2 b+2]);
set(h,'LineWidth',2);
title ('Ellisse')
risposta = input ('Digita 1 per scegliere manualmente il campione
t.\nDigita 2 per scegliere il campione in modo random.\nFai la tua scelta:
');
if (risposta == 1)
      j = input('Scegliere l''indice del campione t compreso tra 1 e il
massimo numero di campioni immesso n:');
else
      j = 1+round(rand*(n-1)); % Genera un intero casuale compreso
                               % tra 1 e n
      disp ('Il campione scelto ha indice');
      disp(j);
end

z_t0 = Zk(j); % Salva in z_t0 il j-simo punto della circonferenza


% Visualizziamo il punto fissato sull'ellisse
plot(real(z_t0),imag(z_t0),'sk')

% L'istruzione calcola i rapporti incrementali relativi ai punti generati.
```

```
% diff(Zk) rappresenta la misura del segmento tra 2 campioni
% successivi, mentre diff(Tk) l'incremento
z1t = diff(Zk)./diff(Tk);

% Rapporto incrementale relativo al campione considerato
% Esso costituirà il coefficiente angolare della retta tangente
z1_t0 = z1t(j);

% Scriviamo l'equazione della retta tangente alla circonferenza nel
% particolare punto
% [-1 2]Sostituisce t. Dato che per disegnare una retta occorrono
% due punti, è come se avessimo assegnato due valori al parametro t
zTang = z_t0+z1_t0*[-1 2];

% Disegnamo la retta tangente
h1 = plot(zTang);
axis equal;
set(h1,'LineWidth',2,'Color','k');
title('Ellisse');

end % end function
```

```
%
```

```
%   Matematica Applicata e Computazionale
%
%   Funzioni complesse di variabili reali - Liv 1
%   per punti campione
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%

%%%%%%%%%%%%%%%%%%%%
% Funzione Parabola
%%%%%%%%%%%%%%%%%%%%

function parabola

clc % Pulisci la schermmata MATLAB


disp('Hai scelto la parabola');
% Scelta del numero di campioni n
n = input('Inserire il numero di campioni:');

% Determinazione di n campioni equispaziati tra -2 e 2
Tk = linspace(-2,2,n);
Zk = Tk+1i*Tk.^2; % Equazione parametrica della parabola
k = 1:length(Zk); % Costruzione della griglia di interpolazione
t=linspace(k(1),k(end),99); % Griglia equispaziata di 99 punti nei
% quali calcolare il valore dell'interpolante
% Interpoliamo la funzione Zk nei punti t sulla griglia k usando la function
% interp1.
% yy = interp1(X, FX, XX, TIPO), dove
% X è la griglia di interpolazione;
```

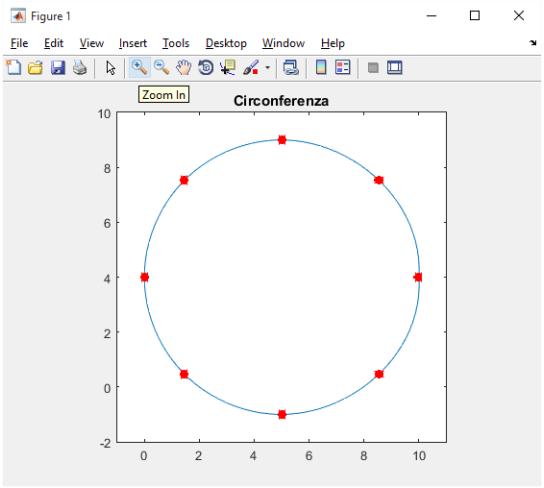
```
% FX sono i valori da interpolare;
% XX sono i punti sui quali si vuole calcolare l'interpolante
% TIPO è il tipo di interpolazione richiesta:
% - 'nearest' : interpolazione con polinomio costante a tratti
% - 'linear' : (default) polinomio lineare a tratti
% - 'cubic' : polinomio cubico a tratti (con derivate continue)
% - 'spline' : interpolazione con spline cubica
Zt = interp1(k,Zk,t,'spline');

plot(Zt); % Disegna la parabola
hold on;
axis equal;
h = plot (Zk,'*r'); % Disegna i campioni
set(h,'LineWidth',2);
title ('Parabola')

risposta = input ('Digita 1 per scegliere manualmente il campione
t.\nDigita 2 per scegliere il campione in modo random.\nFai la tua scelta:
');
if (risposta == 1)
      j = input('Scegliere l''indice del campione t compreso tra 1 e il
massimo numero di campioni immesso n:');
else
      j = 1+round(rand*(n-1)); % Genera un intero casuale compreso tra 1 e
n
      disp ('Il campione scelto ha indice');
      disp(j);
end

z_t0 = Zk(j); % Salva in z_t0 il j-simo punto della parabola

% Visualizziamo il punto fissato sulla parabola
plot(real(z_t0),imag(z_t0),'sk')

% L'istruzione calcola i rapporti incrementali relativi ai punti
% generati. diff(Zk) rappresenta la misura del segmento tra 2 campioni
% successivi, mentre diff(Tk) l'incremento
z1t = diff(Zk)./diff(Tk);

% Rapporto incrementale relativo al campione considerato
% Esso costituirà il coefficiente angolare della retta tangente
z1_t0 = z1t(j);
```

```
% Scriviamo l'equazione della retta tangente alla circonferenza nel
% particolare punto
% [-1 2]Sostituisce t. Dato che per disegnare una retta occorrono
% due punti, è come se avessimo assegnato due valori al parametro t
zTang = z_t0+z1_t0*[-2 2];

% Disegnamo la retta tangente
h1 = plot(zTang);
axis equal;
set(h1,'LineWidth',2,'Color','k')
title('Parabola');

end % end function
```

Esempio d'uso

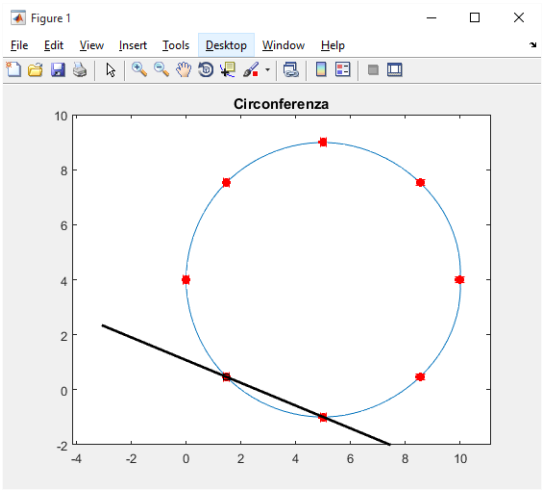
Circonferenza, facciamo partire il programma

```
Hai scelto la circonferenza
Inserire il numero di campioni:9
Inserisci il raggio:5
Inserisci le coordinate del centro:
Ascissa:5
Ordinata:4
Digita 1 per scegliere manualmente il campione t.
Digita 2 per scegliere il campione in modo random.
Fai la tua scelta:
```

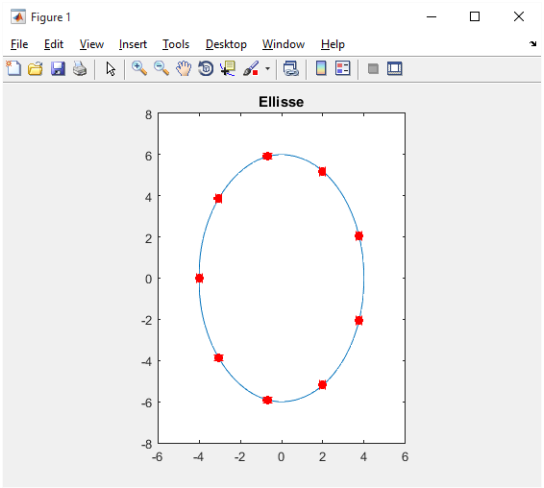



```
Fai la tua scelta: 2
```

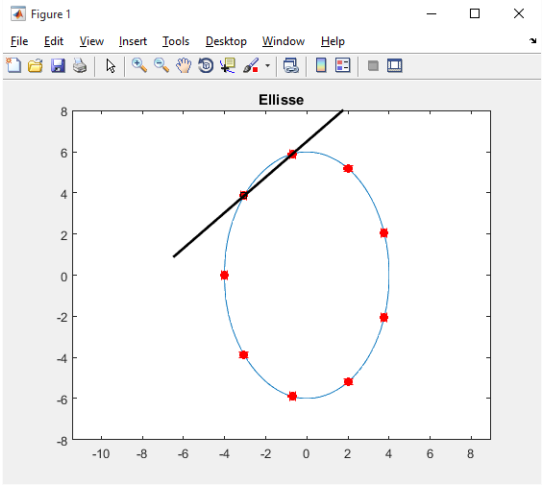
```
Il campione scelto ha indice
     6
```



Ellisse, facciamo partire il programma

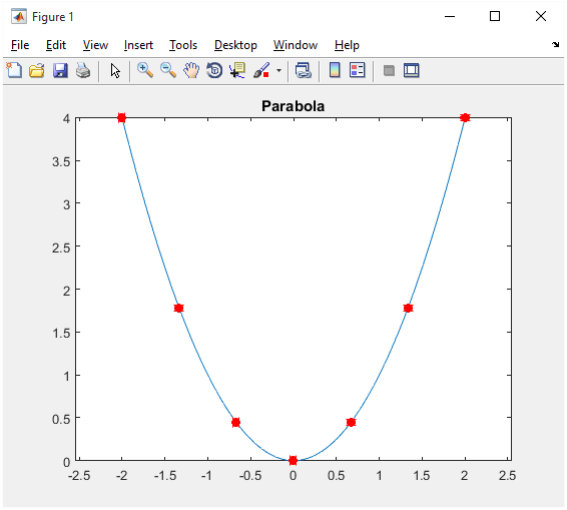



```
Il campione scelto ha indice
```

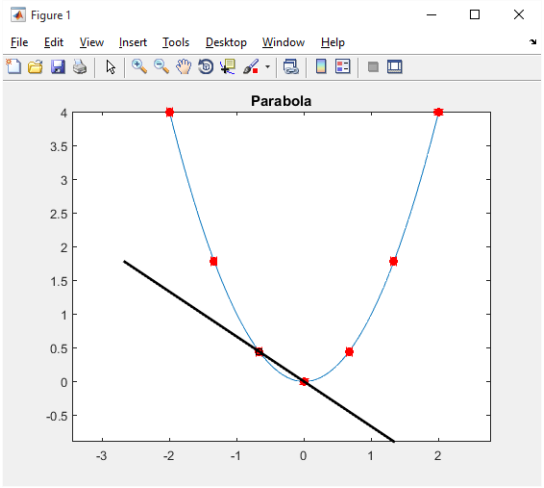


Parabola, facciamo partire il programma

```
Hai scelto la parabola
Inserire il numero di campioni:7
Digita 1 per scegliere manualmente il campione t.
Digita 2 per scegliere il campione in modo random.
Fai la tua scelta:
```




```
Fai la tua scelta: 2
Il campione scelto ha indice
```



ES. 10 – Liv. 1 – Interpolazione Mano

Ricostruire una curva 2D mediante interpolazione: ad esempio, da alcuni campioni ricostruire la propria mano oppure il profilo nei dati contenuti nella cartella `dati_interpolazione`.

Soluzione Proposta

Il seguente programma è composta da

1 Matlab Script Chiamante denominato **interpolazionemano**

2 Function Implementate da me, e cioè **f_interpolazionemano()** e **f_interpolazionemanospline( )**

5 file matrice mxn forniti che vengono caricati mediante la funzione *load()* e scelti dall'utente in fase di avvio del programma

ed effettua l'interpolazione di alcuni Nodi assunti dai file **.mat** forniti in ingresso**,** ove i punti sono memorizzati sotto forma di matrice **mxn** di sole 2 righe, una che rappresenta le Ascisse, l'altra le Ordinate.

Una interessante alterntiva, in assenza delle matrici fornite, è la MATLAB Function

[x,y]=ginput(n_punti)

che restituisce in x e y le coordinate del pixel cliccato sugli assi di una figura Che consente di avere l'input da finestra grafica, calcando l'orma della propria mano appoggiata sul display con una sequenze di click in corrispondenza dei punti che si desidera campionare

```
%   Matematica Applicata e Computazionale
%
%   Interpolazione mano - Liv 1
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                Â© 2016
%
%  GNU/GPL License
%
%
% USO
%
% Lanciare da consolle il file
%
% >> interpolazionemano
%
% e vedere i calcoli che esegue

clc % Pulisci la schermmata MATLAB

% Scelta della curva del piano
scelta = menu ('Scegli la curva', 'Mano', 'Campania', 'Napoli', 'Carpi',
'Ischia');

% Usiamo lo switch per effettuare le scelte
switch scelta
      %%%%%%%%%%%%%%%
      % Case 1 Mano
      %%%%%%%%%%%%%%%
      case 1

        % CARICA ARRY 2D mano CHE CONTIENE COORDINATE DEI PUNTI
        load('mano.mat') % necessita del file mano.mat in locale

        figure % visualizza la figura
```

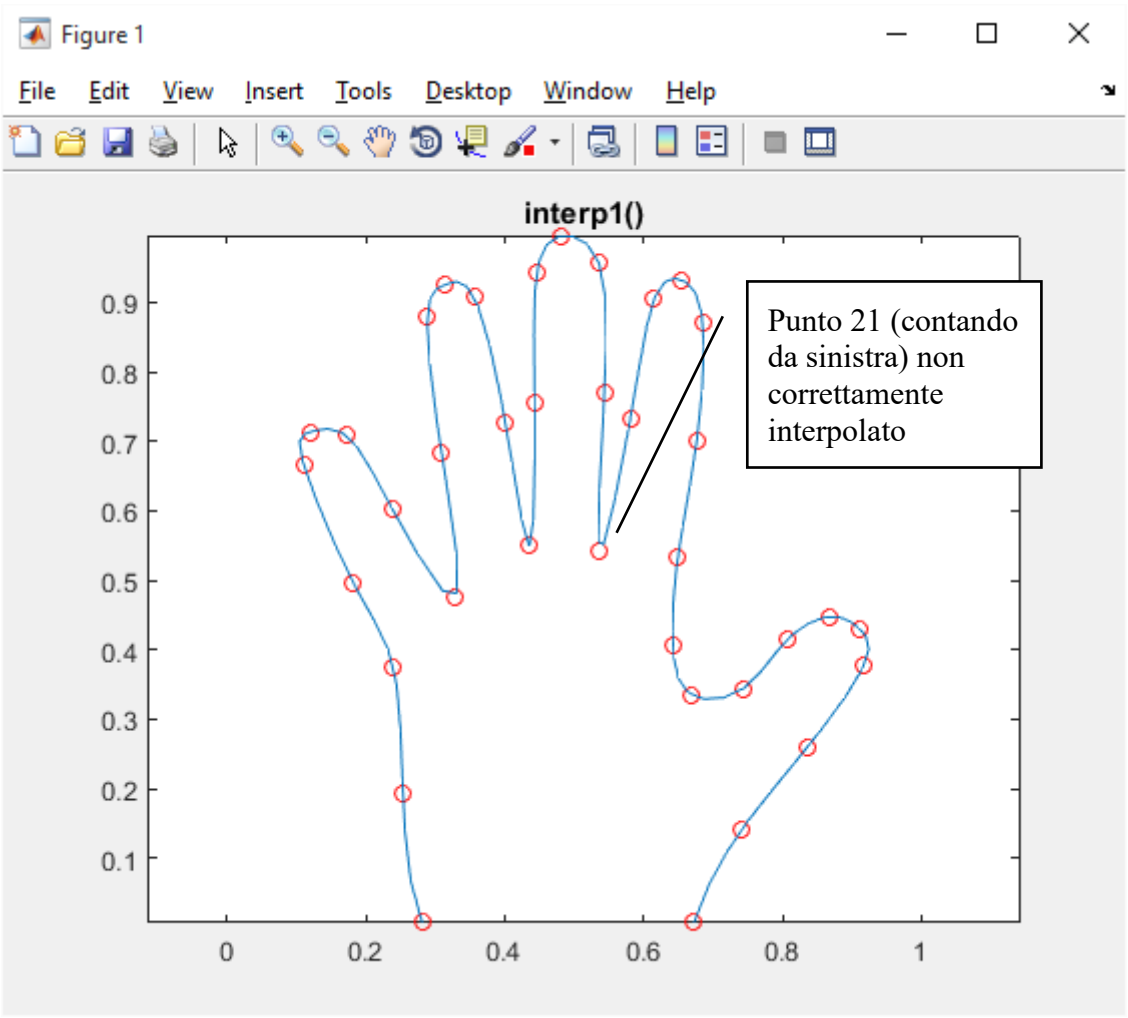
```
    f_interpolazionemano(mano); % INTERPOLAZIONE interp1()

    figure

    f_interpolazionemanospline(mano); % INTERPOLAZIONE CUBICA spline()

%%%%%%%%%%%%%%%%%%
% Case 2 Campania
%%%%%%%%%%%%%%%%%%
case 2

    % CARICA ARRY 2D mano CHE CONTIENE COORDINATE DEI PUNTI
    load('campania.mat') % necessita del file campania.mat in locale

    figure % visualizza la figura

    f_interpolazionemano(campania); % INTERPOLAZIONE interp1()

    figure

    f_interpolazionemanospline(campania); % INTERPOLAZIONE CUBICA
                                          % spline()

%%%%%%%%%%%%%%%%
% Case 3 Napoli
%%%%%%%%%%%%%%%%
case 3

    % CARICA ARRY 2D mano CHE CONTIENE COORDINATE DEI PUNTI
    load('napoli.mat') % necessita del file napoli.mat in locale

    figure % visualizza la figura

    f_interpolazionemano(napoli); % INTERPOLAZIONE interp1()

    figure

    f_interpolazionemanospline(napoli); % INTERPOLAZIONE CUBICA
                                        % spline()
```

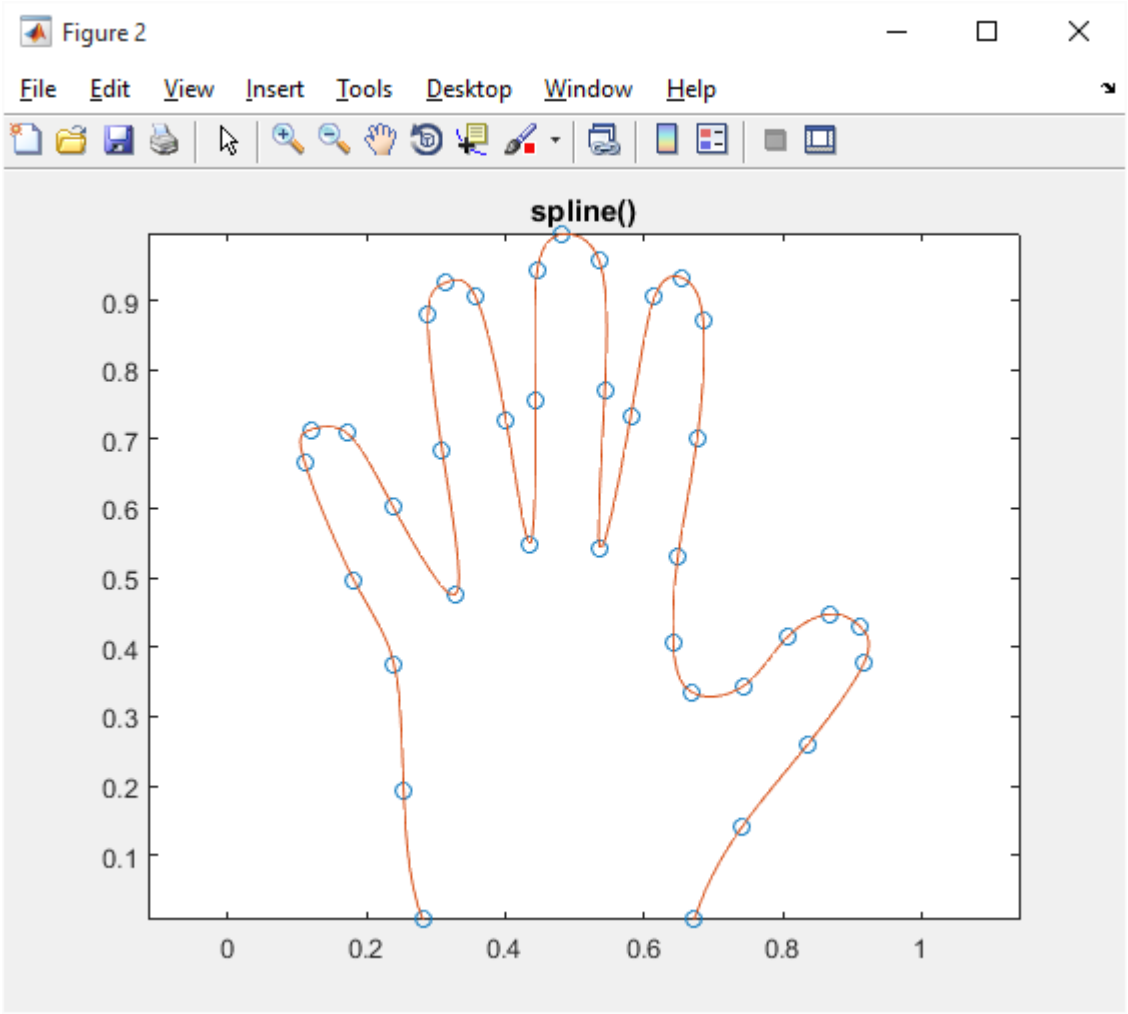
```
%%%%%%%%%%%%%%%
% Case 4 Capri
%%%%%%%%%%%%%%%
case 4

  % CARICA ARRY 2D mano CHE CONTIENE COORDINATE DEI PUNTI
  load('capri.mat') % necessita del file capri.mat in locale

  figure % visualizza la figura

  f_interpolazionemano(capri); % INTERPOLAZIONE interp1()

  figure

  f_interpolazionemanospline(capri); % INTERPOLAZIONE CUBICA
                                     % spline()

%%%%%%%%%%%%%%%%%%
% Case 5 Ischia
%%%%%%%%%%%%%%%%%%
case 5

  % CARICA ARRY 2D mano CHE CONTIENE COORDINATE DEI PUNTI
  load('ischia.mat') % necessita del file ischia.mat in locale

  figure % visualizza la figura

  f_interpolazionemano(ischia); % INTERPOLAZIONE interp1()

  figure

  f_interpolazionemanospline(ischia); % INTERPOLAZIONE CUBICA
                                      % spline()


end % end Switch
```

```
%   Matematica Applicata e Computazionale
%
%   Interpolazione mano - Liv 1
%
%        Programma elaborato da
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION f_interpolazionemano.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% QUIZ: La curva ricostruita con interp1() è sempre regolare?
% RISPOSTA: La Curva Ottenuta con interp1() non è del tutto regolare, anche
%           se la metodologia da essa utilizzata è impostata a spline sulla
%           function Ciò si evince dal 21° punto che non viene toccato
%           dalla curva. Inoltre è appena percettibile in bassa misura,
%           causa Basso Numero di Nodi, l'andamento della Curva che è
%           differente da spline() a interp1()

% INTERPOLAZIONE CUBICA CON interp1() DI TIPO SPLINE
function f_interpolazionemano( mano )
x=mano(:,1); % COLONNA DELLE ASCISSE X
y=mano(:,end); % COLONNA DELLE ORDINATE Y
plot(x,y,'or')
hold on
k=1:length(x);
t=linspace(k(1),k(end),99);

x_interpolati=interp1(k,x,t,'spline'); % INTERPOLAZIONE LUNGO LE X
y_interpolati=interp1(k,y,t,'spline'); % INTERPOLAZIONE LUNGO LE Y

plot(x_interpolati,y_interpolati);
title('interp1()');
axis equal
end % end function
```

```
%
%   Matematica Applica e Computazionale
%
%   Interpolazione mano spline - Liv 1
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION f_interpolazionemanospline.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% INTERPOLAZIONE CUBICA CON SPLINE

function f_interpolazionemanospline( mano )

x=mano(:,1); % COLONNA DELLE ASCISSE X
y=mano(:,end); % COLONNA DELLE ORDINATE Y
t=1:length(x); % GRIGLIA FITTA SU RANGE DELL'INDICE
tt=linspace(t(1),t(end),250);

xt=spline(t,x,tt); % INTERPOLO ASCISSE
yt=spline(t,y,tt); % INTERPOLO ORDINATE
plot(x,y,'o',xt,yt)

title('spline()');

axis equal

end % end function
```

Esempio d’uso: In questo esempio considereremo solo la mano





Quiz: la curva ricostruita con `interp1()` è sempre regolare?

Numero di Nodi, l'andamento della Curva che è differente da `spline()` a `interp1()`

B – Funzioni complesse di variabili complesse

ES. 11 – Liv. 1 – Funzioni complesse di variabili complesse

Per le funzioni [complesse di variabile complessa] elencate sotto, produrre i grafici MATLAB per visualizzarne alcune caratteristiche (continuità, periodicità nel dominio visualizzato):

$$
\begin{array}{cccc}
 & f(z)=e^{z} & f(z)=e^{\frac{1}{z}} & f(z)=\operatorname{Re} z & f(z)=\sqrt{z^{2}-1} \\
(\S) & f(z)=\log z & f(z)=\overline{z} & f(z)=\operatorname{Im} z & f(z)=\dfrac{1}{z^{2}-1} \\
 & f(z)=\sin z & f(z)=\cos z & f(z)=|z| & f(z)=\dfrac{\sin z}{|z|}
\end{array}
$$

Scegliere opportunamente il dominio di visualizzazione ed il tipo di griglia (circolare o rettangolare).

```
%   Matematica Applicata e Computazionale

%   Funzioni complesse di variabili complesse - Liv 1
%
%        Programma elaborato da
%           Giovanni DI CECCA
%               108/1569
%        http://www.dicecca.net
%                © 2016
%  GNU/GPL License
%
% USO
% Lanciare da consolle il file
% >> f_complex_vis
%
% e vedere i calcoli che esegue

clc % Pulisci la schermmata MATLAB

% MAIN
uscita='No';
f_funzioni_complesse_vis(uscita);
close all
```

```
%   Matematica Applicata e Computazionale
%
%   Funzioni complesse di variabili complesse - Liv 1
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION f_funzioni_complesse_vis.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function f_funzioni_complesse_vis(uscita)

while strcmp(uscita,'No')

      % MENU' DI SCELTA FUNZIONE
      sceltaFunzione=menu('Scegli una funzione',...
            'f(z)=exp(z)',...
            'f(z)=exp(1/z)',...
            'f(z)=real(z)',...
            'f(z)=sqrt(z^2-1)',...
            'f(z)=log(z)',...
            'f(z)=conj(z)',...
            'f(z)=imag(z)',...
            'f(z)=abs(z)',...
            'f(z)=1/(z^2-1)',...
            'f(z)=sin(z)',...
            'f(z)=cos(z)',...
            'f(z)=sin(z)/abs(z)');
```

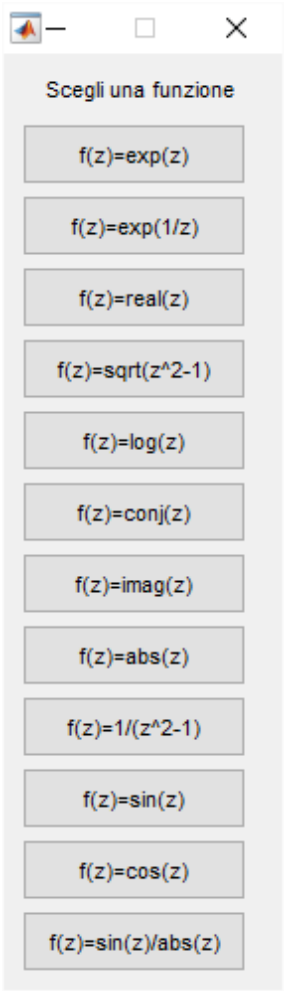
```
% MENU' DI SCELTA GRIGLIA (CIRCOLARE O RETTANGOLARE)
      sceltaGriglia=menu('Scegli il tipo di
griglia','Circolare','Rettangolare');
      titoloInputDialog='Dati di input';
      nomeCampo='w (w>0)';
      default={'0.0'};
      num_linee=1;

      % MENU' DI SCELTA DEL DOMINIO ATTRAVERSO UNA INPUT DIALOG BOX
      w = inputdlg(nomeCampo,titoloInputDialog,num_linee,default);
      w = char(w);
      w = str2num(w);

      switch sceltaGriglia
            case 1
                  z=w*cplxgrid(30);
            case 2
                  [x,y]=meshgrid(linspace(-w,w,30));
                  z=x+1i*y;
      end

      switch sceltaFunzione
            case 1
                  f=exp(z);
            case 2
                  f=exp(1./z);
            case 3
                  f=real(z);
            case 4
                  f=sqrt((z.^2)-1);
            case 5
                  f=log(z);
            case 6
                  f=conj(z);
            case 7
                  f=imag(z);
            case 8
                  f=abs(z);
            case 9
                  f=1./((z.^2)-1);
            case 10
                  f=sin(z);
```

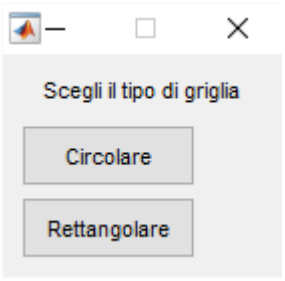
```
            case 11
                  f=cos(z);
            case 12
                  f=sin(z)./abs(z);

            end % END SWITCH

      %DISEGNA GRAFICO FUNZIONI E GRIGLIA
      figure
      surf(real(z), imag(z),real(f))
      hold on; V=axis;
      mesh(real(z),imag(z),V(5)*ones(size(z)))

      % CRITERI DI SELEZIONE USCITA PER GESTIRE CICLO WHILE
      % ATTRAVERSO UNA QUESTION DIALOG BOX
      uscita=questdlg('Vuoi uscire?','Continua...');

end % END WHILE
end % END FUNCTION
```

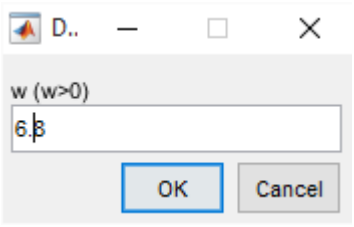
Esempio d'uso

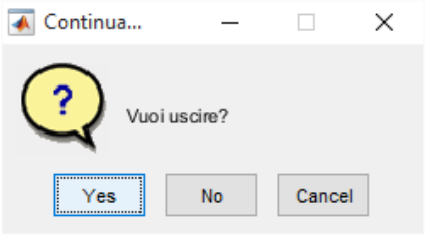
Per non rendere inutilmente lunga la presente documentazione, a titolo esemplificativo di questo esercizio sceglieremo un singolo esempio (cosa che avverrà anche negli altri esercizi che seguiranno), in quanto le funzioni stesse sono già primitive del sistema MATLAB

Consideriamo la funzione **f(z)=conj(z)** e vediamo come questa viene rappresentata sia in griglia circolare che rettangolare.


Scegli una funzione
f(z)=exp(z)
f(z)=exp(1/z)
f(z)=real(z)
f(z)=sqrt(z^2-1)
f(z)=log(z)
f(z)=conj(z)
f(z)=imag(z)
f(z)=abs(z)
f(z)=1/(z^2-1)
f(z)=sin(z)
f(z)=cos(z)
f(z)=sin(z)/abs(z)


Scegli il tipo di griglia
Circolare
Rettangolare


D..
w (w>0)
6.3
OK
Cancel


Continua...
Vuoi uscire?
Yes
No
Cancel

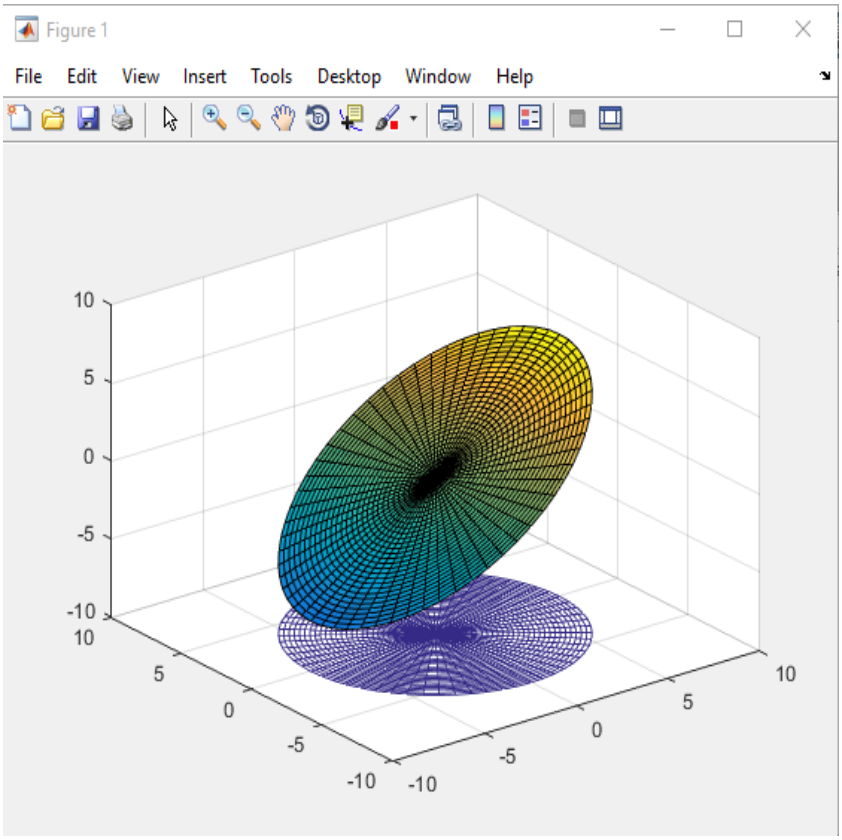
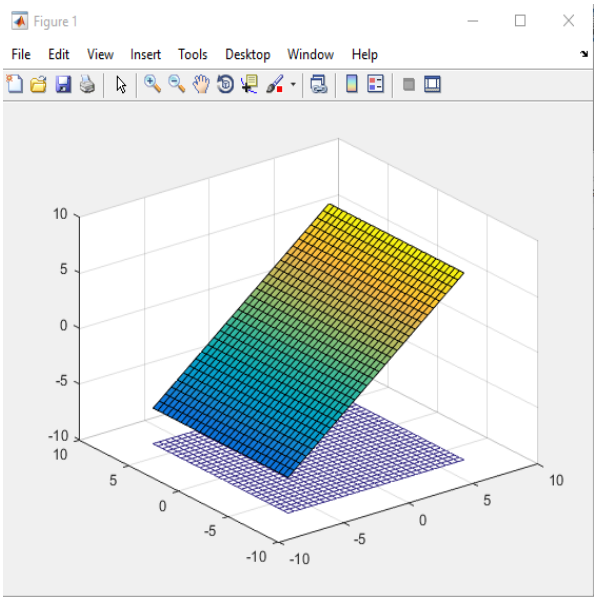
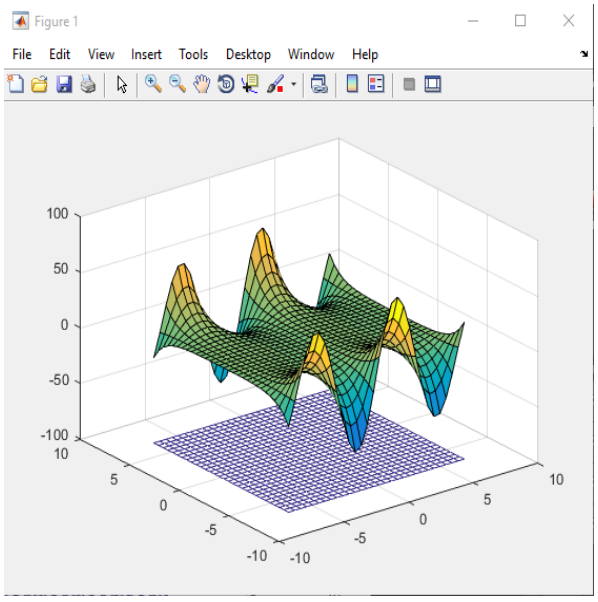
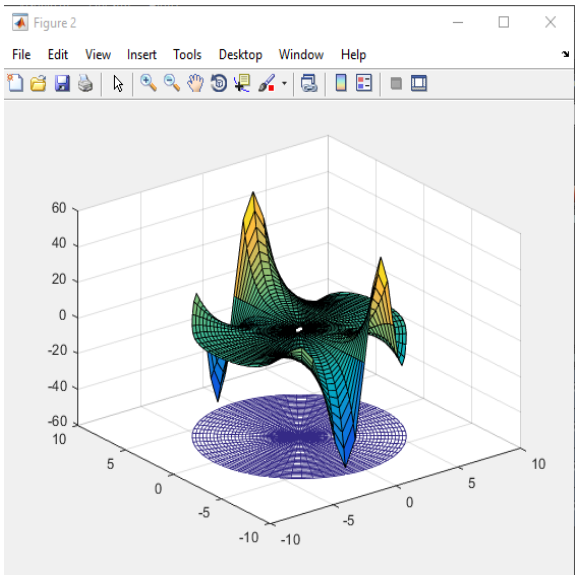


Figure 1
File Edit View Insert Tools Desktop Window Help
10
5
0
-5
-10
10
5
0
-5
-10
-10
-5
0
5
10



Altri esempi (f(z)=sin *z* / abs(*z*))





ES. 12 – Liv. 1 – Funzioni complesse intorno

Per le funzioni complesse (§) visualizzare la continuità o discontinuità in un intorno (circolare e rettangolare) $I(z_0)$ di alcuni punti z_0 del piano complesso.

Soluzione proposta

L' algoritmo, composto da un programma chiamante e da una function, risulta molto simile a quello dell'Esercizio 11 trasportato in Symbolic e modificato in modo tale da assumere in input, oltre alla **Semiampiezza di Intorno w**, anche un valore per il **Punto Complesso z_0** dell'Intorno considerato, verificando al suo interno la Continuità della Funzione Scelta da Menu

```
%   Matematica Applicata e Computazionale
%
%   FUnzioni complesse di variabili complesse - Liv 1
%        Programma elaborato da

%           Giovanni DI CECCA
%               108/1569
%        http://www.dicecca.net
%
%                © 2016
%  GNU/GPL License
% USO
% Lanciare da consolle il file
% >> fcplxintorno
% e vedere i calcoli che esegue

clc % Pulisci la schermmata MATLAB

%%%%%%%
% MAIN
%%%%%%%

uscita='No';
f_funz_cmplx_intorno(uscita); % carica la funzione
close all
```

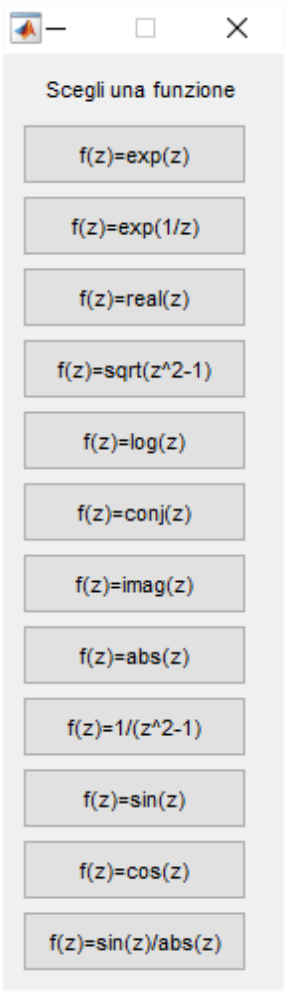
```
%   Matematica Applicata e Computazionale
%
%   Funzioni complesse di variabili complesse - Liv 1
%
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION f_funz_cmplx_intorno.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function f_funz_cmplx_intorno(uscita)

while strcmp(uscita,'No')
      % MENU' DI SCELTA FUNZIONE
      sceltaFunzione=menu('Scegli una funzione',...
            'f(z)=exp(z)',...
            'f(z)=exp(1/z)',...
            'f(z)=real(z)',...
            'f(z)=sqrt(z^2-1)',...
            'f(z)=log(z)',...
            'f(z)=conj(z)',...
            'f(z)=imag(z)',...
            'f(z)=abs(z)',...
            'f(z)=1/(z^2-1)',...
            'f(z)=sin(z)',...
            'f(z)=cos(z)',...
            'f(z)=sin(z)/abs(z)');
```

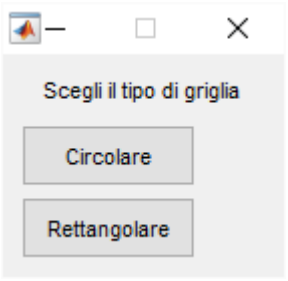
```
	% MENU' DI SCELTA GRIGLIA (CIRCOLARE O RETTANGOLARE)
	sceltaGriglia=menu('Scegli il tipo di
griglia','Circolare','Rettangolare');

	% MENU' SCELTA PUNTO DI ACCUMULAZIONE z0
	% E DELLA SEMIAMPIEZZA DI INTORNO w (raggio dell'intorno)
	campi={'z0','w (w>0)'};
	titoloDLG='Input di Dati';
	numeroRighe=1;
	valoriDefault={'i','0.1'};
	datiInput = inputdlg(campi,titoloDLG,numeroRighe,valoriDefault);
	datiInput = char(datiInput);
	datiInput = str2num(datiInput);
	z0 = datiInput(1);
	w = datiInput(2);

	switch sceltaGriglia
		case 1
			% la funzione cplxgrid crea una griglia circolare di
centro
			% 0 e raggio 1.Con la seguente istruzione trasliamo il
centro
			% della griglia in z0 e ne modifichiamo il raggio, che
da 1
			% assume il valore w dato in input
			z=z0+(w*cplxgrid(30));
		case 2
			% creazione di una griglia rettangolare che poi diventa
di
			% centro z0
			[x,y]=meshgrid(linspace(-w,w,30));
			z=z0+(x+1i*y);
		end

	switch sceltaFunzione
		case 1
			f=exp(z);
		case 2
			f=exp(1./z);
		case 3
			f=real(z);
		case 4
```

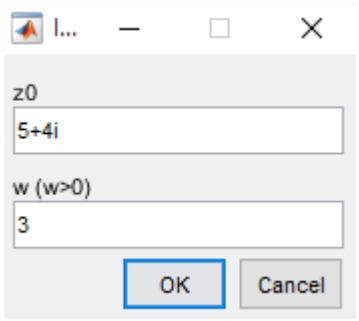
```
                f=sqrt((z.^2)-1);
            case 5
                f=log(z);
            case 6
                f=conj(z);
            case 7
                f=imag(z);
            case 8
                f=abs(z);
            case 9
                f=1./((z.^2)-1);
            case 10
                f=sin(z);
            case 11
                f=cos(z);
            case 12
                f=sin(z)./abs(z);
        end % END SWITCH

        % DISEGNA GRAFICO FUNZIONI E GRIGLIA
        figure
        surf(real(z), imag(z),real(f))
        hold on;
        V=axis;
        mesh(real(z),imag(z),V(5)*ones(size(z)))

        % ASSE PERPENDICOLARE AL PUNTO DI ACCUMULAZIONE E PASSANTE PER z0
        % PUO' AIUTARCI PER UNA QUESTIONE DI VISIBILITA'
        plot3(real(z0)*ones(1,2),imag(z0)*ones(1,2),V(5:6),...
        '-k.','LineWidth',4,'MarkerSize',36);

        %CRITERI DI SELEZIONE USCITA PER GESTIRE CICLO WHILE
        %ATTRAVERSO UNA QUESTION DIALOG BOX
        uscita=questdlg('Vuoi uscire?','Continua...');

end % END WHILE

end % END FUNCTION
```

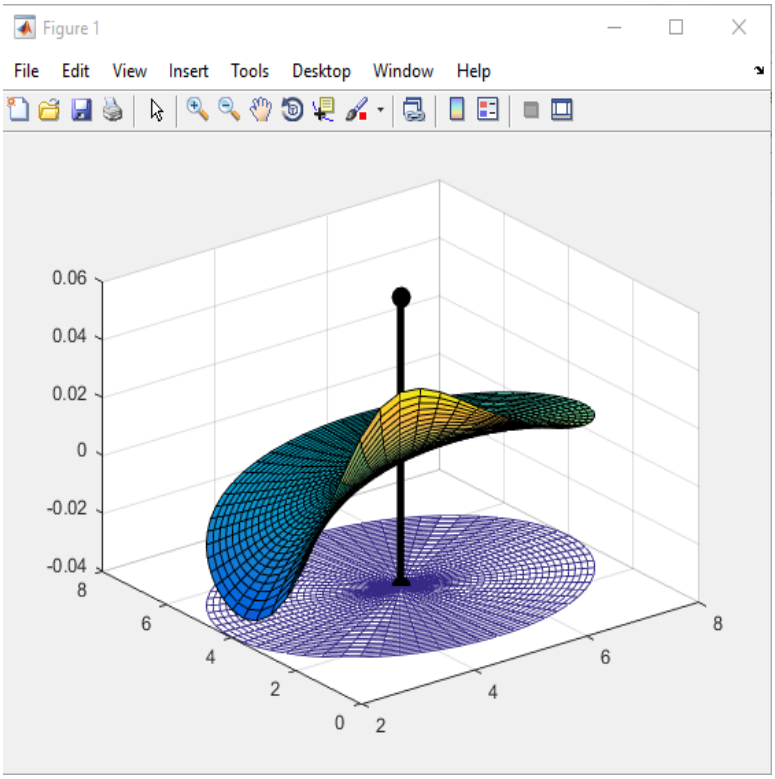
Esempio d'uso

Come nel caso precedente abbiamo considerato un singolo esempio, ovvero la funzione f(z)=1/(z^2-1)









ES. 13 – Liv. 1 – Funzioni complesse di variabili complesse Numerico/Simbolico

Per le funzioni complesse (§) verificare e visualizzare la continuità o discontinuità in un intorno $I(z_0)$ di alcuni punti z_0 del piano complesso mediante Symbolic Math Toolbox.

Soluzione proposta

La parte Numerica si occupa dell'aspetto Grafico, mentre la parte Simbolica lavora sull'aspetto Algoritmico della Continuità.

L'Algoritmo infatti calcola il Limite della Funzione Complessa nel Punto di accumulazione sym_z0 Complesso, ovvero il Gemello Simbolico del punto z0 dato in Input, opportunamente traslato. Il Limite è in Forma Polare, in modo da poter calcolare tutto per sostituzione di 'rho->0'.

Dopo aver calcolato il Limite, l'Algoritmo "Tenta" di Calcolare il Valore della Funzione in sym_z0 f(sym_z0) entrando in un **TRY CATCH**, che serve a gestire il caso Infinito di un'eventuale Divisione per 0, a causa della quale la Matlab Function subs() restituisce un Errore, in tal caso il CATCH assegna alla Variabile del risultato la Stringa "Inf".

Con tali risultati è possibile Verificare la Continuità della Funzione in sym_z0, sulla logica del fatto che una Funzione si Definisce Continua in un punto se

- Esiste nel Punto z0 f(z0)=y
- Esiste il suo Limite in z0 e coincide con f(z0) lim(f(z0))=f(z0)

Dagli Output Grafici, possiamo poi visualizzare le zone di Continuità e quelle di Discontinuità sulla Curva

```
%   Matematica Applicata e Computazionale
%
%   Funzioni complesse di variabili complesse - Liv 1
%   Simbolico e numerico
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
% USO
%
% Lanciare da consolle il file
%
% >> symfcplx_intorno
%
% e vedere i calcoli che esegue

clc % Pulisci la schermata MATLAB

%%%%%%%
% MAIN
%%%%%%%

uscita='No';
fsym_funz_cmplx_intorno(uscita);
close
```

```
%   Matematica Applicata e Computazionale
%
%   Funzioni complesse di variabili complesse - Liv 1
%   Simbolico e numerico
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION fsym_funz_cmplx_intorno.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function fsym_funz_cmplx_intorno(uscita)

syms rho positive

syms theta real;

while strcmp(uscita,'No')
      % MENU' DI SCELTA FUNZIONE
      sceltaFunzione=menu('Scegli una funzione',...
            'f(z)=exp(z)',...
            'f(z)=exp(1/z)',...
            'f(z)=real(z)',...
            'f(z)=sqrt(z^2-1)',...
            'f(z)=log(z)',...
            'f(z)=conj(z)',...
            'f(z)=imag(z)',...
            'f(z)=abs(z)',...
            'f(z)=1/(z^2-1)',...
            'f(z)=sin(z)',...
            'f(z)=cos(z)',...
```

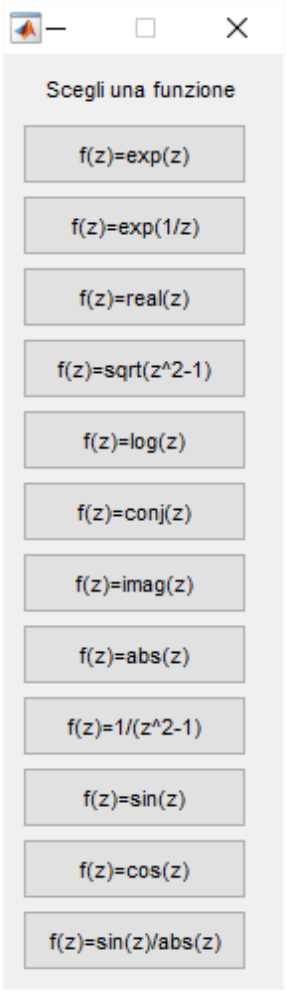
```
            'f(z)=sin(z)/abs(z)');

      % MENU' DI SCELTA GRIGLIA (CIRCOLARE O RETTANGOLARE)
      sceltaGriglia=menu('Scegli il tipo di
griglia','Circolare','Rettangolare');

      % MENU' DI SCELTA DEL PUNTO DI ACCUMULAZIONE z0 E DELLA SEMIAMPIEZZA
DI INTORNO w
      campi={'z0','w (w>0)'};
      titoloDLG='Input di Dati';
      numeroRighe=1;
      valoriDefault={'0','0.1'};
      datiInput = inputdlg(campi,titoloDLG,numeroRighe,valoriDefault);
      datiInput = char(datiInput);
      datiInput = str2num(datiInput);
      z0=datiInput(1);
      w=datiInput(2);
      z0_sym=z0+rho*exp(1i*theta);  % TRANSLIAMO  z0  NUMERICO  A  z0_sym
SIMBOLICO

      switch sceltaGriglia
            case 1
                  % la funzione cplxgrid crea una griglia circolare di
% centro
                  % 0 e raggio 1.Con la seguente istruzione trasliamo il
% centro
                  % della griglia in z0 e ne modifichiamo il raggio, che
% da 1
                  % assume il valore w dato in input
                  z=z0+(w*cplxgrid(30));
            case 2
                  % creazione di una griglia rettangolare che poi diventa
% di
                  % centro z0
                  % CREO UN'INTORNO DI z0 DEFINENDO PRIMA IL DOMINIO DI
% ESISTENZA
                  % DEI VETTORI z NELL'AMPIEZZA [-w,w] POI TRANSLO L'ARRAY
% z A z0
                  [x,y]=meshgrid(linspace(-w,w,30));
                  z=z0+(x+1i*y); % z0: traslazione - x+iy:omotetia
      end % END SWITCH
```

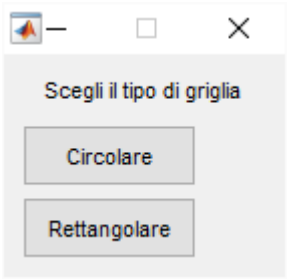
```
% trasla z0 in z0_sym, che servirà poi per calcolare il
% limite ( simbolico
z0_sym=z0+rho*exp(1i*theta);

switch sceltaFunzione
      case 1
            f=exp(z);
            fsym=exp(z0_sym);
      case 2
            f=exp(1./z);
            fsym=exp(1/z0_sym);
      case 3
            f=real(z);
            fsym=real(z0_sym);
      case 4
            f=sqrt((z.^2)-1);
            fsym=sqrt((z0_sym^2)-1);
      case 5
            f=log(z);
            fsym=log(z0_sym);
      case 6
            f=conj(z);
            fsym=conj(z0_sym);
      case 7
            f=imag(z);
            fsym=imag(z0_sym);
      case 8
            f=abs(z);
            fsym=abs(z0_sym);
      case 9
            f=1./((z.^2)-1);
            fsym=1/((z0_sym^2)-1);
      case 10
            f=sin(z);
            fsym=sin(z0_sym);
      case 11
            f=cos(z);
            fsym=cos(z0_sym);
      case 12
            f=sin(z)./abs(z);
            fsym=sin(z0_sym)/abs(z0_sym);
end % END SWITCH
```

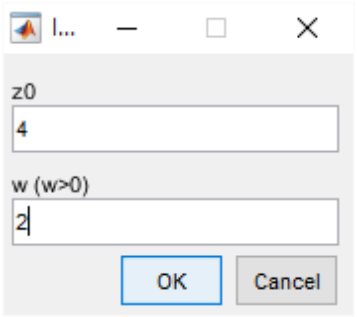
```
      % DISEGNA GRAFICO FUNZIONI E GRIGLIA
      figure
      surf(real(z), imag(z),real(f))
      hold on; V=axis;
      mesh(real(z),imag(z),V(5)*ones(size(z)))
      plot3(real(z0)*ones(1,2),imag(z0)*ones(1,2),V(5:6),...
      '-k.','LineWidth',4,'MarkerSize',36);

      % ASSE PERPENDICOLARE PERPENDICOLARE A z0
      % PUO' AIUTARCI PER UNA QUESTIONE DI VISIBILITA'
      % CALCOLO L'ESISTENZA DELLA FUNZIONE f(z0_sym) NELL'INTORNO DI 0
      % POICHE' HO GIA' TRANSLATO z0_sym A z0 E POSSO CALCOLARE TUTTO IN
      % FORMA POLARE PER rho->0.
      limite=limit(fsym,rho,0);

      % TRY CATCH
      % SERVE A GESTIRE IL CASO IN CUI LA subs() RITORNI UNA DIVISIONE
      % PER 0 CHE PROVOCA UN ERRORE IN MATLAB
      try % PROVA A EFFETTUARE IL CALCOLO
            fsym_z0=subs(fsym,rho,0); % CALCOLO IL VALORE DELLA FUNZIONE
%IN z0
      catch % ALTRIMENTI ASSEGNA Inf
            fsym_z0='Inf';
      end % END TRY CATCH

      % STAMPO SU CONSOLE I RISULTATI
      display(['[ f(z0) ','limite(f(z0) ]']);
      display([fsym_z0, limite])
      display('Se i due risultati sono UGUALI');
      display('Allora la Funzione f(z) e'' CONTINUA!');

      % QUESTION DIALOG BOX PER LA SCELTA DI USCITA
      uscita=questdlg('Vuoi uscire?','Continua...');

end % END WHILE

end % END FUNCTION
```

Esempio d'uso

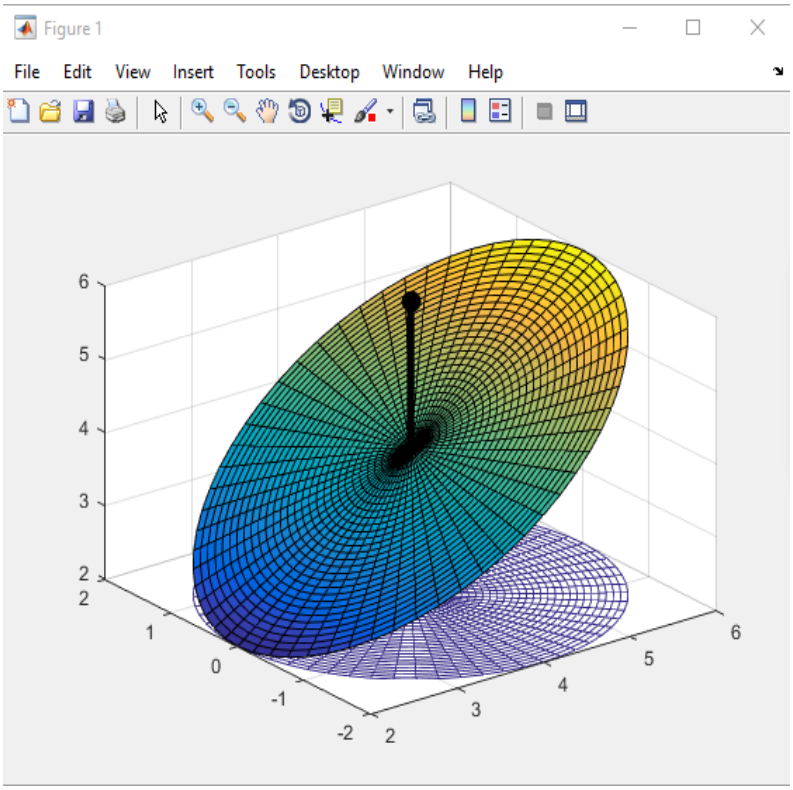
Come nei casi precedenti consideriamo un solo caso, ad esempio **f(z)=conj(z)**








```
[ f(z0) limite(f(z0) ]

ans =

[ 4, 4]

Se i due risultati
sono UGUALI
Allora la Funzione
f(z) e' CONTINUA!
[ f(z0) limite(f(z0) ]

ans =

[ 4, 4]

Se i due risultati
sono UGUALI
Allora la Funzione
f(z) e' CONTINUA!
>>
```



ES. 14 – Liv - 1 - Funzione circolare di z_0

Creare e visualizzare, mediante le funzioni `ezsurf()`, `ezmesh()` del Symbolic Math Toolbox, un intorno circolare del punto z_0 , assegnato in input con la dimensione massima dell'intorno.

Soluzione proposta

L'esercizio è simile a quello precedente

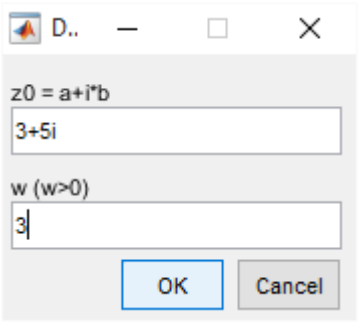
```
%   Matematica Applicata e Computazionale
%
%   FUnzioni complesse di variabili complesse - Liv 1
%   Funzione circolare di z0
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
% USO
%
% Lanciare da consolle il file
%
% >> sym_circolare
%
% e vedere i calcoli che esegue

clear; clc; close all % Pulisci la schermmata e memoria di MATLAB

%%%%%%%
% MAIN
%%%%%%%

syms z % Simbolico di MATLAB
```

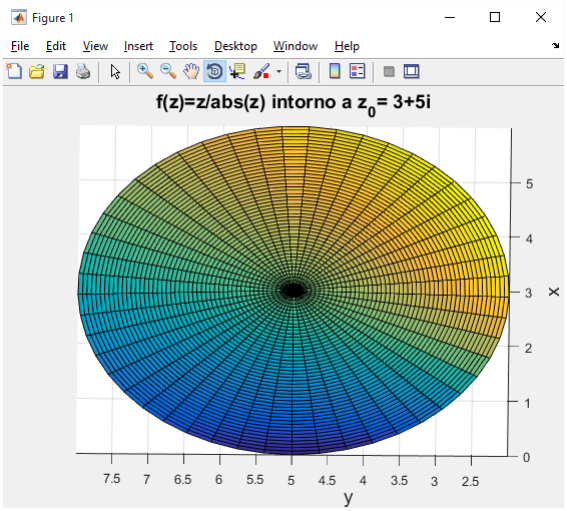
```
fz = scelta_funzioni; % sceglie la funzione (carica da apposita function)

% Sceglie z0 e la semiampiezza o il raggio
% Valori di default w=0.5; z0=i;

% Input dati dal Symbolic Toolbox
dlg_title='Dati di input'; prompt={'z0 = a+i*b','w (w>0)'};
num_lines=1; default={'i','0.5'};
answer = inputdlg(prompt,dlg_title,num_lines,default);
answer = char(answer);
answer = str2num(answer); % converti i valori in numerico

% Valori calcolati
z0 = answer(1);
w = answer(2);


% GRAFICI (intorno circolare)
% usa le coordinate polari [rh,th]
syms rh positive; syms th real;
Fz=subs(fz,z,z0+rh*exp(1i*th));

x=real(z0)+rh*cos(th); % parte reale
y=imag(z0)+rh*sin(th); % parte immaginaria

% intorno circolare C
C=[0 w -pi pi];


figure % Stampa figura
% DISEGNO DEL MODULO DI fz
ezsurf(x,y,real(Fz),C); hold on; axis tight; V=axis;
h=plot3(real(z0)*ones(1,2),imag(z0)*ones(1,2),V(5:6),'-k.');
set(h,'LineWidth',3,'MarkerSize',36)
xlabel('x','FontSize',14);
ylabel('y','FontSize',14);
zlabel('real[f(z)]','FontSize',14);
title(['f(z)=' char(fz) ' intorno a z_0= ' num2str(z0)],'FontSize',14);
```

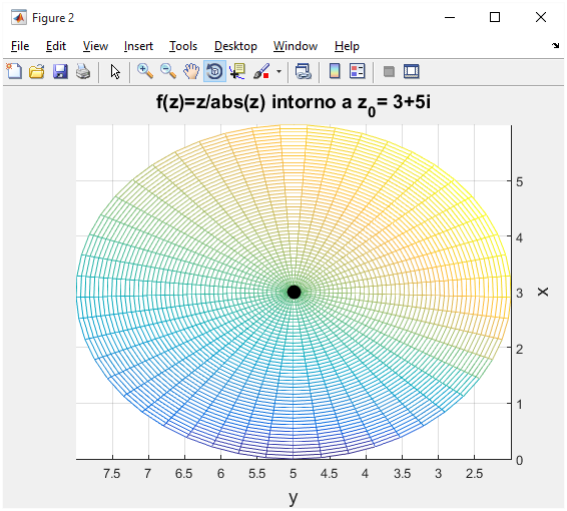
```
figure % Stampa figura
% DISEGNO DEL MODULO DI fz in 3D
ezmesh(x,y,real(Fz),C); hold on; axis tight; V=axis;
h=plot3(real(z0)*ones(1,2),imag(z0)*ones(1,2),V(5:6),'-k.');
set(h,'LineWidth',3,'MarkerSize',36)
xlabel('x','FontSize',14);
ylabel('y','FontSize',14);
zlabel('real[f(z)]','FontSize',14)
title(['f(z)=' char(fz) ' intorno a z_0= ' num2str(z0)],'FontSize',14)
```

```
%   Matematica Applicata e Computazionale
%
%   FUnzioni complesse di variabili complesse - Liv 1
%   Funzione circolare di z0
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function: scelta_funzioni.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function fz = scelta_funzioni

% FUNZIONI TEST
cellFun={'z^2';            % 1) funzione continua
         'exp(z)';         % 2) funzione continua
         'sqrt(z)';        % 4) funzione non continua per z=x<0
         'z/abs(z)';       % 5) funzione continua tranne che in z0=0
         '(z-i)/abs(z-i)'; % 6) funzione continua tranne che in z0=i
    };

k = menu('scegli funzione f(z)',cellFun);
fz = sym(cellFun{k}); % Visualizza il modulo di
return
```

Esempio d’uso







Funzioni Olomorfe

ES. 15 – Liv. 1 – Funzioni Olomorfe

Visualizzare l'olomorfia (o non) delle funzioni (§) in un intorno (circolare e rettangolare) $I(z_0)$ di alcuni punti z_0 del piano complesso.

```
%   Matematica Applicata e Computazionale
%
%   Funzioni Olomorfe - Liv 1
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
% USO
%
% Lanciare da consolle il file
%
% >> vis_olomorfia
%
% e vedere i calcoli che esegue

%%%%%%%
% MAIN
%%%%%%%

clc % Pulisci la schermmata MATLAB

uscita='No';
f_vis_olomorfia(uscita);
close all
```

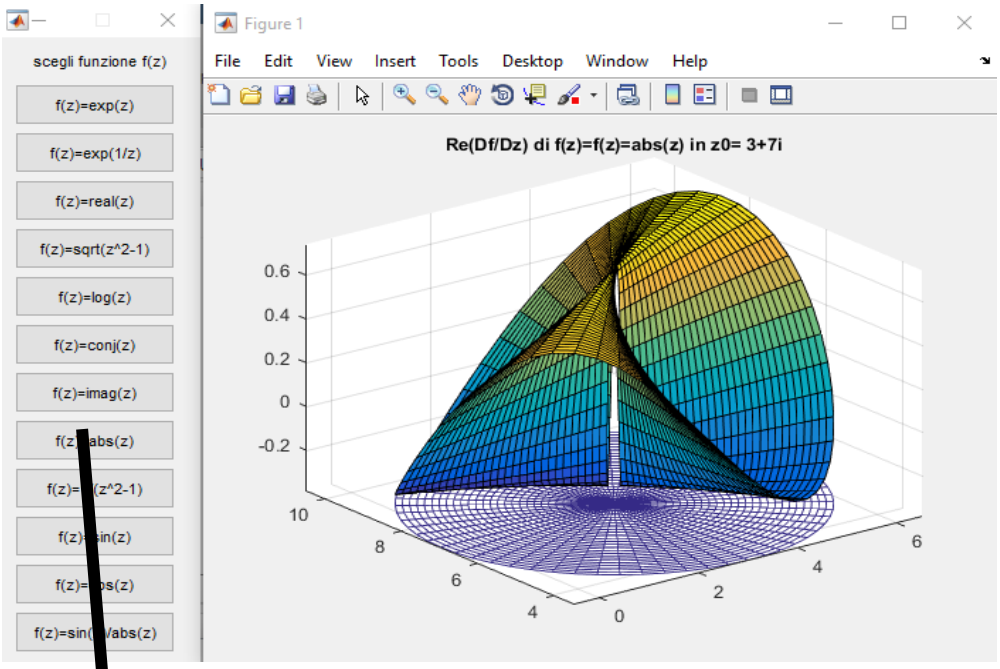
```
%   Matematica Applica e Computazionale
%   Funzioni Olomorfe - Liv 1
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUnction: f_vis_olomorfia.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function f_vis_olomorfia(uscita)

while strcmp(uscita,'No')

      % RICHIAMA LA FUNCTION PER LA SELEZIONE DELLA FUNZIONE DA DISEGNARE
      [funzione, fStringa] = scelta_funzione();
      sceltaGriglia=menu('Scegli il tipo di
griglia','Circolare','Rettangolare');

      % MENU' DI SCELTA DEL PUNTO z0 E DELLA SEMIAMPIEZZA DI INTORNO w
      campi={'z0','w (w>0)'};
      titoloDLG='Input di Dati';
      numeroRighe=1;
      valoriDefault={'i','0.1'};
      datiInput = inputdlg(campi,titoloDLG,numeroRighe,valoriDefault);
      datiInput = char(datiInput);
      datiInput = str2num(datiInput);
      z0 = datiInput(1);
      w = datiInput(2);

      switch sceltaGriglia
            case 1
                  % crea griglia circolare di centro z0 e raggio w, con 30
punti
```

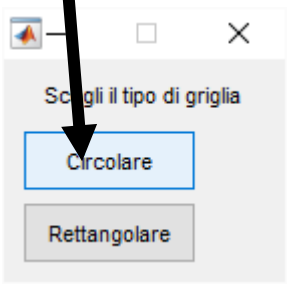
```
                z=z0+(w*cplxgrid(30));
            case 2
                % crea griglia rettangolare di centro z0 e 30 campioni
compresi tra
                % -w e w
                [x,y]=meshgrid(linspace(-w,w,30));
            z=z0+(x+1i*y);
        end

        fz=funzione(z); % VAOLRE DELLA FUNZOIONE IN z
        f0=funzione(z0); % VAOLRE DELLA FUNZOIONE IN z0
        Dz=z-z0; % INCREMENTO DELLA VARIABILE DIPENDENTE
        Df=fz-f0; % INCREMENTO DEL VALORE DELLA FUNZIONE
        DfDz=Df./Dz; % RAPPORTO INCREMENTALE

        % DISEGNA LA PARTE REALE DEL RAPPORTO INCREMENTALE.ESSA CI
        % FORNISCE INFORMAZIONI PIU' ATTENDIBILI RISPETTO ALLA PARTE
IMMAGINARIA
        figure
        surf(real(z),imag(z),real(DfDz));
        title(['Re(Df/Dz) di f(z)=' fStringa ' in z0= '
num2str(z0)],'FontSize',10)
        hold on; axis tight; V=axis;
        mesh(real(z),imag(z),V(5)*ones(size(z)))

        % DISEGNA LA PARTE IMMAGINARIA DEL RAPPORTO INCREMENTALE. CON
        % L'ALTERAZIONE DEL SUO ANDAMENTO PUO ORIENTARCI A LIVELLO INTUITIVO
        % NELLA STIMA DELLA DERIVABILITA' MA AD OCCHIO NON FORNISCE
        % CONCRETAMENTE INFORMAZIONI UTILI
        figure
        surf(real(z),imag(z),imag(DfDz));
        title(['Im(Df/Dz) di' fStringa ' in z0= ' num2str(z0)],'FontSize',10)
        hold on; axis tight; V=axis;
        mesh(real(z),imag(z),V(5)*ones(size(z)))

        % CRITERI DI SELEZIONE USCITA PER GESTIRE CICLO WHILE
        % ATTRAVERSO UNA QUESTION DIALOG BOX
        uscita=questdlg('Vuoi uscire?','Continua...');

end % END WHILE

end % END FUNCTION
```

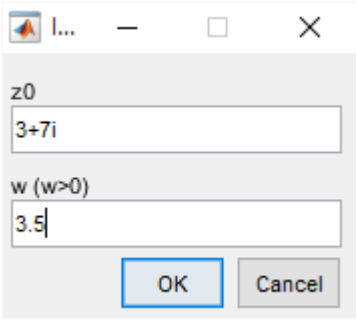
```
%
%   Matematica Applica e Computazionale
%
%   Funzioni Olomorfe - Liv 1
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function: scelta_funzione.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [f, fString] = scelta_funzione()

% FUNZIONI TEST
cellFunzioni={'f(z)=exp(z)'; 'f(z)=exp(1/z)'; 'f(z)=real(z)';
'f(z)=sqrt(z^2-1)';'f(z)=log(z)';'f(z)=conj(z)';
'f(z)=imag(z)';'f(z)=abs(z)';'f(z)=1/(z^2-1)';
'f(z)=sin(z)';'f(z)=cos(z)';'f(z)=sin(z)/abs(z)';};

funzioneScelta = menu('scegli funzione f(z)',cellFunzioni);

switch funzioneScelta
      case 1
            f=@(z) exp(z);

      case 2
            f=@(z) exp(1./z);

      case 3
            f=@(z) real(z);
```

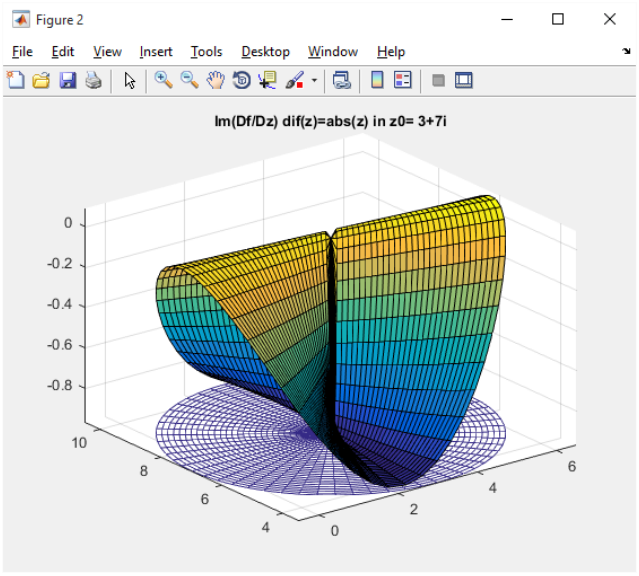
```
        case 4
              f=@(z) sqrt((z.^2)-1);

        case 5
              f=@(z) log(z);

        case 6
              f=@(z) conj(z);

        case 7
              f=@(z) imag(z);

        case 8
              f=@(z) abs(z);

        case 9
              f=@(z) 1/((z.^2)-1);

        case 10
              f=@(z) sin(z);

        case 11
              f=@(z) cos(z);

        case 12
              f=@(z) sin(z)./abs(z);

        end % END SWITCH

        fString=cellFunzioni{funzioneScelta};

end % END Function
```

Esempio d'uso









Come si può notare osservando i grafici, ed in particolare quello relativo alla parte reale del rapporto incrementale, la funzione non è derivabile nel punto considerato, e ciò viene mostrato dall'andamento irregolare delle superfici.

Rammentiamo infatti che una funzione è olomorfa in un punto se esiste ed è finito il limite in senso complesso del suo rapporto incrementale.

Graficamente, l'esistenza di tale limite si deduce dalla presenza degli stessi colori attorno al punto considerato, e ciò non accade nel nostro esempio

ES. 16 – Liv. 1 – Funzioni Olomorfe – Versione Simbolica

Verificare e visualizzare la derivabilità in senso complesso delle funzioni (§) mediante Symbolic Math Toolbox

Soluzione Proposta

L'esercizio è equivalente a quello precedente ma viene usato il calcolo simbolico.

La differenza è che l'Olomorfia è stimabile in termini di Calcoli ed i risultati sono stampati come Stringhe in text() a Finestra Grafica.

Nel caso in cui la Funzione non sia Derivabile nel punto z0, il programma lancia una Error Dialog Box dopo aver controllato il tutto attraverso il TRY CATCH.

In caso Contrario stampa ad Axis il Grafico

NOTA: per selezionare la funzione su cui operare viene usata di nuovo la function **`scelta_funzione()`** vista nel precedente elaborato

```
%
%   Matematica Applicata e Computazionale
%
%   Funzioni Olomorfe - Liv 1
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
% USO
%
% Lanciare da consolle il file
%
```

```
% >> sym_vis_olomorfia
%
% e vedere i calcoli che esegue

%%%%%%%
% MAIN
%%%%%%%

clc % Pulisci la schermmata MATLAB

uscita='No';
fsym_vis_olomorfia(uscita);
close all
```

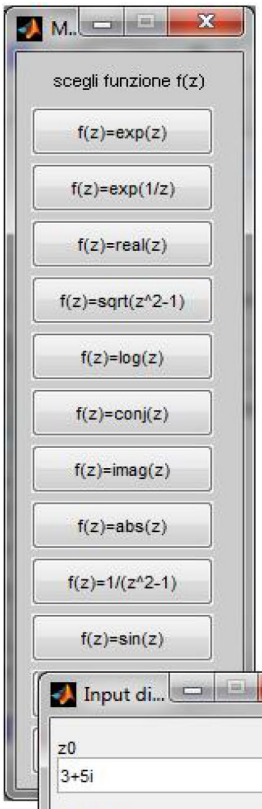
```
%   Matematica Applica e Computazionale
%
%   Funzioni Olomorfe - Liv 1
%   Versione Symbolica
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function: fsym_vis_olomorfia.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function fsym_vis_olomorfia(uscita)

syms x y Dx Dy teta real;

syms ro positive;

z=x+1i*y;

while strcmp(uscita,'No')
      % RICHIAMA LA FUNCTION PER LA SELEZIONE DELLA FUNZIONE DA DISEGNARE
      [funzione, fStringa] = scelta_funzione();

      fz=funzione(z);

      % MENU' DI SCELTA DEL PUNTO z0 E DELLA SEMIAMPIEZZA DI IN-TORNO w
      campi={'z0','w (w>0)'}
      titoloDLG='Input di Dati';
      numeroRighe=1;
      valoriDefault={'i','0.1'};
      datiInput = inputdlg(campi,titoloDLG,numeroRighe,valoriDefault);
```

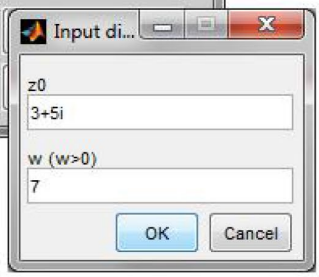
```
datiInput = char(datiInput);
datiInput = str2num(datiInput);
z0 = datiInput(1);
w = datiInput(2);

% INCREMENTO DELLA VARIABILE COMPLESSA CALCOLATO IN FUNZIONE
% DELL'INCREMENTO REALE E DELL'INCREMENTO IMMAGINARIO
Dz=Dx+1i*Dy;

% INCREMENTO Df DELLA FUNZIONE CALCOLATO PER SOSTITUZIONE.
% occorre per poter poi scrivere il rapporto incrementale
Df=subs(fz,{'x','y'},{'x+Dx','y+Dy'})-fz;
R=Df/Dz; % RAPPORTO INCREMENTALE

% RAPPORTO INCREMENTALE IN COORDINATE POLARI
% tale sostituzione è necessaria per poter poi calcolare il
% limite in senso complesso
R_Pol=subs(R,{'Dx','Dy'},{'ro*cos(teta)','ro*sin(teta)'});

% DERIVATA DELLA FUNZIONE (SERVE PER DISEGNARE IL GRAFICO)
derivata_fz=limit(R_Pol,ro,0);

% DERIVATA DELLA FUNZIONE CALCOLATA NEL PUNTO DI ACCUMULAZIONE z0
derivata_fz0=limit(subs(R_Pol,{'x','y'},{real(z0),imag(z0)}),ro,0);

% TRY-CATCH
% LA DERIVATA derivata_fz0 E' UNA STRINGA, NUMERICA O ALFABETICA A
% SECONDA DEL VALORE DEL LIMITE.
% NEL PRIMO CASO LA CONVERSIONE IN DOUBLE double(derivata_z0)
% NON GENERA ERRORI E PERMETTE IL CALCOLO NUMERICO DEL RISULTATO;
% NEL CASO DI STRINGA ALFANUMERICA, LA DERIVATA E' INCALCOLABILE
% POICHE' IL LIMITE DIPENDENTE DA teta E GENERERA' UN ERRORE.
% IL COSTRUTTO try-catch EFFETTUA PROPRIO QUESTO TIPO DI CONTROLLO,
% VISUALIZZANDO UNA FINESTRA DI ERRORE QUANDO IL LIMITE DIPENDA DA
% teta
try % PROVA...
      %... A CONVERTIRE L'EVENTUALE STRINGA ALFANUMERICA
      derivata_fz0=double(derivata_fz0);
      figure
      %...A DISEGNARE LA PARTE REALE DEL LIMITE
      ezsurf(real(derivata_fz),[-w,w]);
      title(['Re(Df/Dz) di ' fStringa ' in z0= '
```

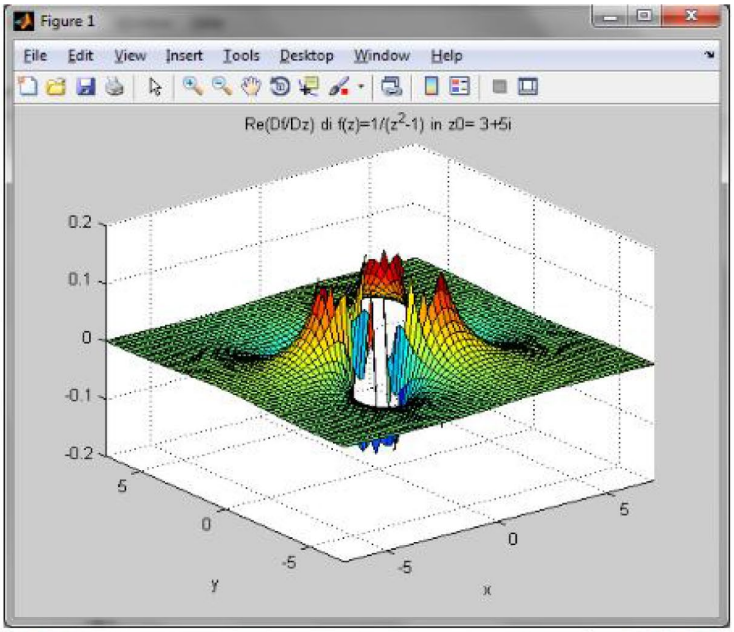
```
            num2str(z0)],'FontSize',10)
            figure
            %...A DISEGNARE LA PARTE IMMAGINARIA DEL LIMITE
            ezsurf(imag(derivata_fz),[-w,w]);
            title(['Im(Df/Dz) di ' fStringa ' in z0= '
            num2str(z0)],'FontSize',10)
            stringaRisultato=['Funzione Derivabile in z0 =' num2str(z0)
...
            ' Derivata in z0 f''(z0)=' num2str(derivata_fz0)];
            figure
            text(.5,.5,stringaRisultato,'FontSize',10,...

      'HorizontalAlignment','center','Color','b','FontWeight','bold');
            axis off
      catch %...IN CASO DI ERRORE, CATTURA VISUALIZZANDO ERROR DIALOG BOX
            errordlg('La Funzione non Ammette Limite');
      end % END TRY CATCH

      % CRITERI DI SELEZIONE USCITA PER GESTIRE CICLO WHILE
      % ATTRAVERSO UNA QUESTION DIALOG BOX
      uscita=questdlg('Vuoi uscire?','Continua...');
end % END WHILE

end % END Function
```

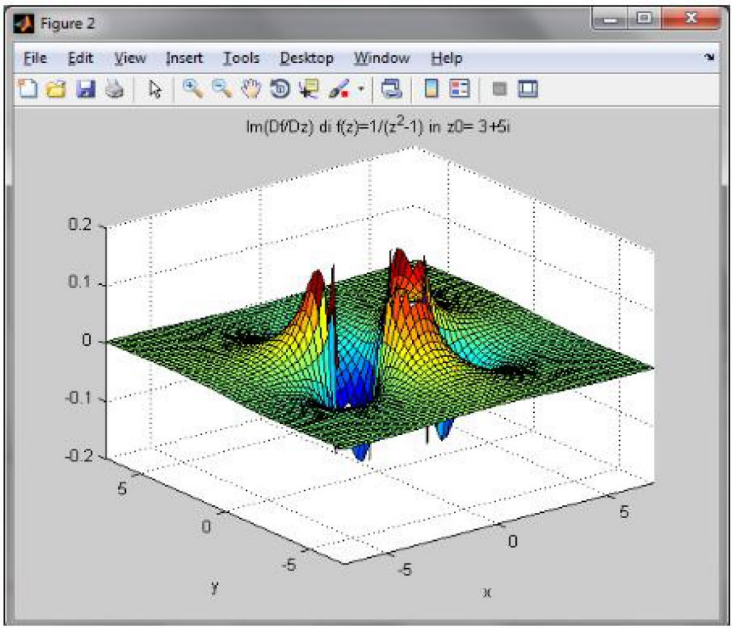
La funzione scelta_funzione.m è la stessa vista nel precedente esercizio e quindi non verrà replicata

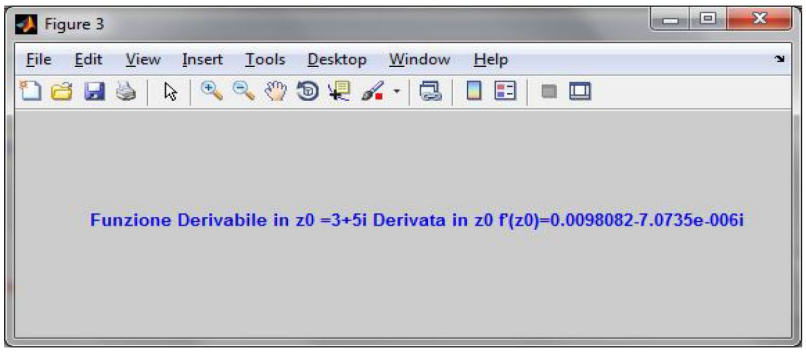
Esempio d'uso











ES. 17 – Liv. 1 – Analisi del limite complesso

Come si calcola mediante il *Symbolic Math Toolbox* di MATLAB il seguente limite in senso complesso

$$f'(z_0) = \lim_{z \to z_0} \frac{f(z) - f(z_0)}{z - z_0} = \lim_{\Delta z \to 0} \frac{f(z_0 + \Delta z) - f(z_0)}{\Delta z}$$

Soluzione Proposta

L'Elaborato calcola la derivata di una funzione complessa come Limite del Rapporto Incrementale, usando la sua forma Polare.

E' da notare che per poter assumere la Funzione da Input Dialog Box, l'unica MATLAB Function che valutasse effettivamente la Stringa Nome raccolta è risultata essere `eval()`.

Per il resto, dopo aver effettuato i classici calcoli sull'Incremento della Funzione e l'Incremento della Variabile Complessa Dfz/Dz, si calcola il Limite del Rapporto Incrementale, ottenuto in Coordinate Polari per sostituzione, e a questo si sostituisce il Valore di z_0 per simulare la tendenza al valore che esso contiene.

Tutta la Parte Matematica dell'Algoritmo è gestita da un Costrutto TRY CATCH, per poter gestire l'unico caso in cui il Calcolo non va buon fine (generando un Errore), ovvero quando la Forma del Limite è Indeterminata.

L'Output consiste in una Finestra Grafica di tipo `text()` che stampa la Stringa con il Valore z_0 a cui Tende il Punto di accumulazione e la Derivata Risultato

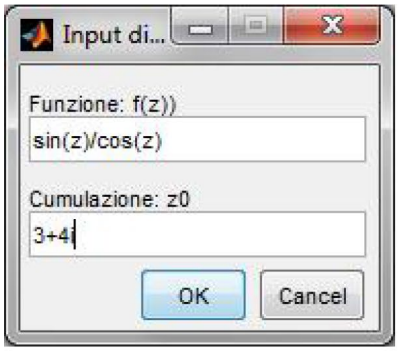
```
%   Matematica Applicata e Computazionale
%
%   Analisi del limite complesso - Liv 1
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
% USO
%
% Lanciare da consolle il file
%
% >> sym_limite
%
% e vedere i calcoli che esegue

%%%%%%%
% MAIN
%%%%%%%

clc % Pulisci la schermmata MATLAB

uscita='No';
while strcmp(uscita,'No')

      % MENU' DI SCELTA DEL PUNTO z0 E DELLA SEMIAMPIEZZA DI INTORNO w
      campi={'Funzione: f(z))','Accumulazione: z0'};
      titoloDLG='Input di Dati';
      numeroRighe=1;
      valoriDefault={'z','i'};

      datiInput = inputdlg(campi,titoloDLG,numeroRighe,valoriDefault);
      datiInput = char(datiInput);
      f = datiInput(1,:);
```

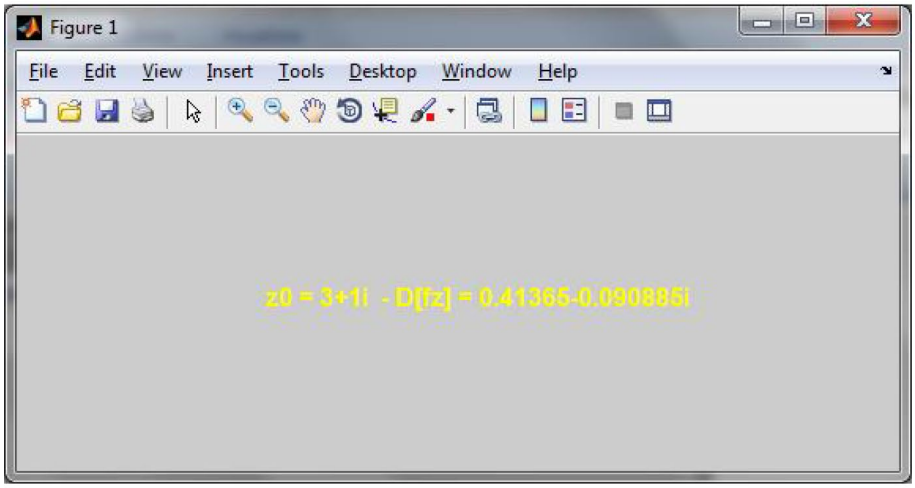
```
        z0 = str2num(datiInput(2,:));
        fsym_limite(f,z0);

        % CRITERI DI SELEZIONE USCITA PER GESTIRE CICLO WHILE
        % ATTRAVERSO UNA QUESTION DIALOG BOX
        uscita=questdlg('Vuoi uscire?','Continua...');

end % END WHILE

close all
```

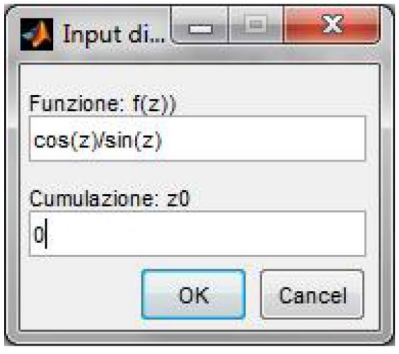
```
%   Matematica Applicata e Computazionale
%
%   Analisi del limite complesso - Liv 1
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%

%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function: fsym_limite.m
%%%%%%%%%%%%%%%%%%%%%%%%%%


function fsym_limite(funzione,z0)

syms x y Dx Dy teta real;

syms ro positive;

z=x+1i*y;
% ATTENZIONE!! IN QUESTO CASO SOLO eval() SEMBRA AVERE EFFETTO SULLA
% CREAZIONE DI funzione, A DIFFERENZA DI inline() O DI UNA ANONYMOUS
% FUNCTION f=@(z)
fz=eval(funzione);

% INCREMENTO DELLA VARIABILE COMPLESSA CALCOLATO IN FUNZIONE
% DELL'INCREMENTO REALE E DELL'INCREMENTO IMMAGINARIO
Dz=Dx+1i*Dy;

% INCREMENTO DELLA FUNZIONE CALCOLATO PER SOSTITUZIONE
% occorre per poter poi scrivere il rapporto incrementale
Df=subs(fz,{'x','y'},{'x+Dx','y+Dy'})-fz;
R=Df/Dz; % RAPPORTO INCREMENTALE
```

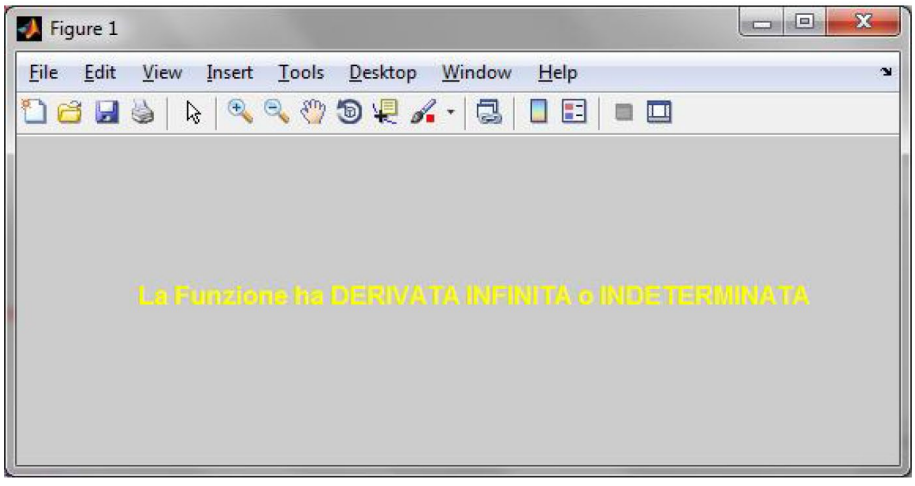
```
try
      % RAPPORTO INCREMENTALE IN COORDINATE POLARI
      R_Pol=subs(R,{'Dx','Dy'},{'ro*cos(teta)','ro*sin(teta)'});
      % DERIVATA GENERICA DELLA FUNZIONE (SERVE PER DISEGNARE IL GRAFICO)
      derivata_fz=limit(R_Pol,ro,0);
      % DERIVATA DELLA FUNZIONE CALCOLATA NEL PUNTO DI CUMULAZIONE z0 DEL
      % LIMITE
      derivata_fz0=limit(subs(R_Pol,{'x','y'},{real(z0),imag(z0)}),ro,0);
      derivata_fz0=double(derivata_fz0);
      z0=num2str(z0);
      stringDerivata=['D[fz] = ' num2str(derivata_fz0)];

catch
      stringDerivata='Limite INFINITO o INDETERMINATO ';
end % End TRY

% Disegna la figura
figure
axis off
text(.5,.5,stringDerivata,'FontSize',12,...
'HorizontalAlignment','center','Color','y','FontWeight','bold');

end % END TRY
```

Esempio d'USO









ES. 18 – Liv. 1 – Analisi del limite complesso

Modificare i codici nella cartella SIMBOLICO di Lab_funz_continue (scaricabile dalla pagina del corso nella sottosezione Funzioni complesse continue) per verificare la derivabilità in un punto.

Genera un errore nel file originale

usa la forma esponenziale di z-z0 = rh*exp(i*th) per
f(z) =
(- i + rh*exp(th*1i) + (3 + 5i))/abs(- i + rh*exp(th*1i) + (3 + 5i))

limit(E,rh,0) =
- ((imag(i) - 5)*1i)/((imag(i) - 5)^2 + (real(i) - 3)^2)^(1/2) - (real(i) - 3)/((imag(i) - 5)^2 + (real(i) - 3)^2)^(1/2)

double(subs(fz,z,z0)) =
Error using symengine (line 59)
DOUBLE cannot convert the input expression into a double array.
If the input expression contains a symbolic variable, use VPA.

Error in sym/double (line 583)
    Xstr = mupadmex('symobj::double', S.s, 0);

Error in limite_fz (line 24)
disp('double(subs(fz,z,z0)) ='); disp(double(subs(subs(fz,z,z0))))

ES. 19 – Liv. 1 – Cauchy-Riemann

Verificare e visualizzare la validità delle Equazioni di Cauchy-Riemann, in senso complesso, per le funzioni (§) in un intorno $I(z_0)$ di alcuni punti z_0 del piano complesso mediante Symbolic Math Toolbox.

Soluzione Proposta

Il Programma verifica l'Olomorfia delle Funzioni Complesse analizzate anche nei precedenti esercizi, usando le Equazioni di Cauchy – Riemman, uno strumento che consente di caratterizzare il campo di olomorfia.

In particolare effettuiamo la Verifica Reale (Piu' Attendibile in MATLAB) e la Verifica Complessa (Meno Attendibile in MATLAB), prendendo in input un Numero Complesso z0 che è il valore a cui tende il Δz del Limite del Rapporto Incrementale (Derivata).

Il Primo Output restituito, Visualizza Graficamente l'intera Olomorfia in un intorno del Valore a cui Tende il Limite, z0, e su tale Grafico valgono le solite regole sulla Stima ad Occhio riguardo l'Andamento delle Curve.

Il Secondo Output è invece la Verifica Matematica sia Reale che Complessa del calcolo del Limite in z0, ed è stampata sotto forma di Stringhe di Testo in una Finestra text()

```
%   Matematica Applica e Computazionale
%
%   Analisi del limite complesso - Liv 1
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
%
% USO
%
% Lanciare da consolle il file
%
% >> sym_cauchyriemann
%
% e vedere i calcoli che esegue

%%%%%%%
% MAIN
%%%%%%%

uscita='No';

while strcmp(uscita,'No')
      syms z0 real

      % MENU' DI SCELTA DEL PUNTO DI CUMULAZIONE z0
    campi={'Cumulazione: z0'};
      titoloDLG='Input di Dati';
      numeroRighe=1;
      valoriDefault={'i'};
      datiInput = inputdlg(campi,titoloDLG,numeroRighe,valoriDefault);
      datiInput = char(datiInput);
      z0 = str2num(datiInput);
```

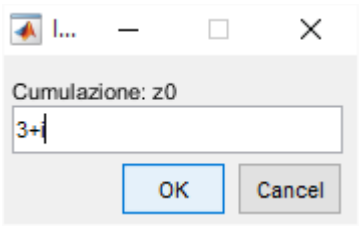
```
        fsym_cauchyriemann(z0);

        % CRITERI DI SELEZIONE USCITA PER GESTIRE CICLO WHILE
        % ATTRAVERSO UNA QUESTION DIALOG BOX
        uscita=questdlg('Vuoi uscire?','Continua...');
end
```

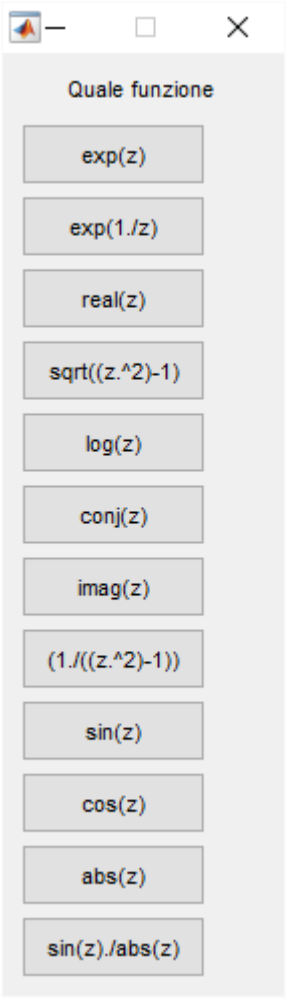
```
%   Matematica Applica e Computazionale
%
%   Analisi del limite complesso - Liv 1
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function: fsym_cauchyriemann.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function fsym_cauchyriemann(z0)

syms x y real;

z=x+1i*y;

funcel={'exp(z)';'exp(1./z)';'real(z)';'sqrt((z.^2)-1)';
'log(z)';'conj(z)'; 'imag(z)';'(1./((z.^2)-1))';
'sin(z)';'cos(z)';'abs(z)';'sin(z)./abs(z)'};

k=menu('Quale funzione',funcel);

fun=simplify(eval(char(funcel{k})));
% APPLICO CAUCHY-RIEMANN PRIMA IN FORMA COMPLESSA POI IN FORMA REALE
% PER VERIFICARE LA VALIDITA' DELLE EQUAZIONI DEL SUO TEOREMA
% NELL'IPOTESI CHE, SIA IN VIA SYMBOLICHE CHE NUMERICA
% IL RISULTATO IN FORMA COMPLESSA, SPESSO, PUO' ESSERE ERRATO
% MENTRE IL CALCOLO IN FORMA REALE E' SEMPRE CORRETTO
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  --- CAUCHY-RIEMANN (FORMA COMPLESSA) ---
% VERIFICA DELL'OLOMORFIA IN FORMA COMPLESSA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% la forma complessa delle equazioni di Cauchy-Riemann è
% df/dx+i*df/dy=0 (EQUAZIONE PRINCIPALE)
%
CRcomp=diff(fun,x)+1i*diff(fun,y);

% SOSTITUZIONE IN CR DI VARIABILI x E y CON PARTE REALE E IMMAGINARIA DI
% z0 per verificarne la validità nel punto z0 fornito in input
CRcomp_z0=subs(CRcomp,{x,y},{real(z0), imag(z0)});

if CRcomp_z0==0 % ottenere 0 significa che è olomorfa
      risultatoComp=' Olomorfa';
else
      risultatoComp=' Non Olomorfa';
end

%  --- CAUCHY-RIEMANN (FORMA REALE) ---
% VERIFICA DELL'OLOMORFIA IN FORMA REALE
% PER VERIFICARE LE EQUAZIONI Useremo la matrice Jacobiana, che
% memorizza in ciascuna riga le derivate parziali di u e v
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FORMA SCALARE DELLE EQUAZIONI DI CAUCHY RIEMANN
% du/dx=dv/dy
% du/dy=dv/dx
%
u=simplify(real(fun)); % PARTE REALE DELLA FUNZIONE
v=simplify(imag(fun)); % PARTE IMMAGINARIA DELLA FUNZIONE

% costruisce la matrice jacobiana di u e v rispetto a x e y
J=jacobian([u,v], [x,y]);

% SOSTITUZIONE CON z0
JJ=subs(J,{x,y},{real(z0), imag(z0)});

% VERIFICA OLOMORFIA SU COMPONENTI DELLE DIAGONALI
verificaReale=(JJ(1,1)==JJ(2,2)) && (JJ(1,2)+JJ(2,1)==0);

if(verificaReale) % se otteniamo 1 ...
      risultatoReale=' Olomorfa';
```

```
else % se otteniamo 0 ...
      risultatoReale=' Non Olomorfa';
end

% TRY CATCH
% QUANDO LA FUNZIONE NON E' OLOMORFA E' IMPOSSIBILE DISEGNARE IL MODULO
% DELLA DIFFERENZA FRA LE COMPONENTI SULLA DIAGONALE PRINCIPALE, E QUINDI
% abs() GENERA UN ERRORE, CHE POSSIAMO SFRUTTARE CATTTURANDOLO PER
% DETERMINARE SE LA FUNZIONE SIA OLOMORFA O MENO
try
      ezsurf(simplify(abs(J(1,1)-J(2,2)))) % NON OLOMORFA IN z0
catch
      ezsurf(J(1,1)-J(2,2)) % SEMPRE OLOMORFA, QUINDI ANCHE IN z0
end % END TRY

title(char(funcel{k}));
figure
cellRisultato={['Verifica Complessa in z0= ' num2str(z0) ' : ' ...
risultatoComp ' (Poco Attendibile in Matlab)'],...
['Verifica Reale in z0= ' num2str(z0) ' : ' ...
risultatoReale ' (Attendibile in Matlab)']};
text(.5,.5,cellRisultato,'FontSize',12,...
'HorizontalAlignment','center','Color','b','FontWeight','bold');
axis off

end % end function
```

Esempio d’uso




Quale funzione
exp(z)
exp(1./z)
real(z)
sqrt((z.^2)-1)
log(z)
conj(z)
imag(z)
(1./((z.^2)-1))
sin(z)
cos(z)
abs(z)
sin(z)./abs(z)

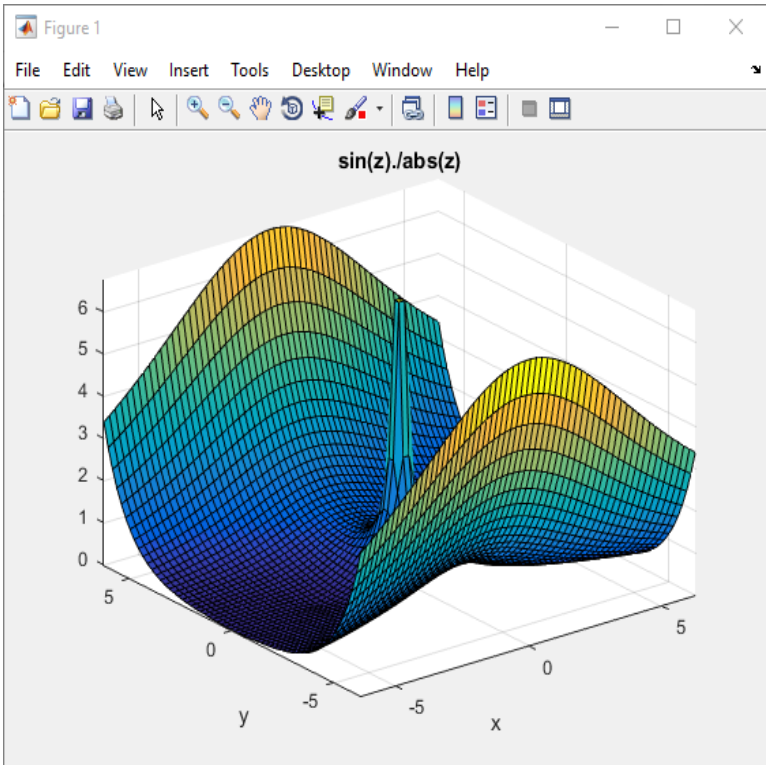


Figure 1
File Edit View Insert Tools Desktop Window Help
sin(z)./abs(z)
6
5
4
3
2
1
0
5
0
-5
y
-5
0
5
x

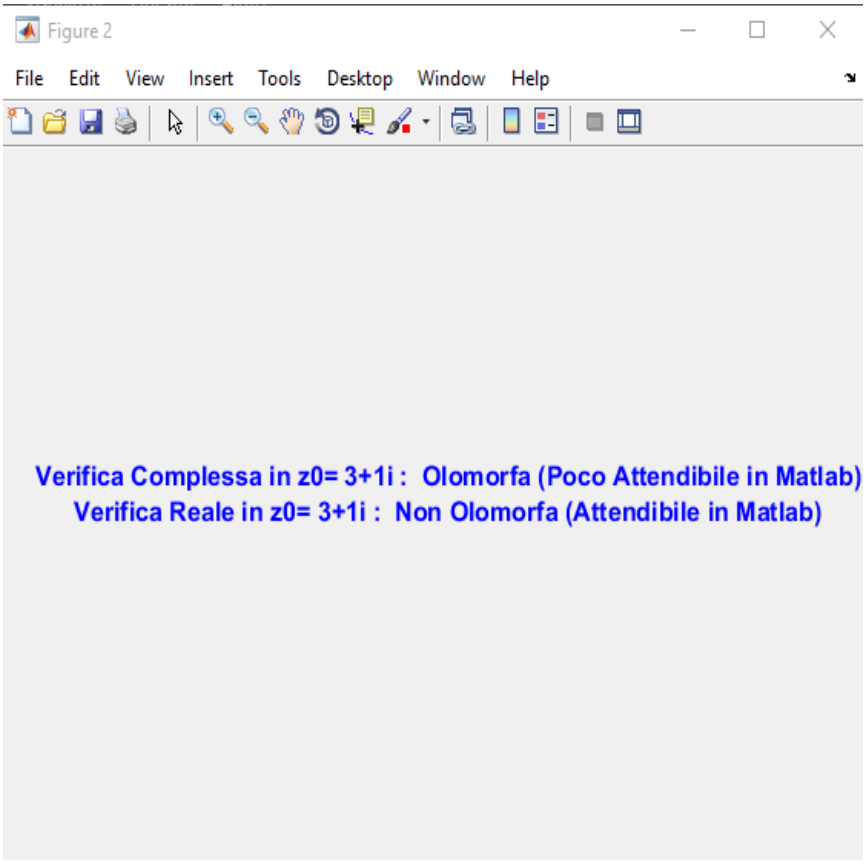


Figure 2
File Edit View Insert Tools Desktop Window Help
Verifica Complessa in z0= 3+1i : Olomorfa (Poco Attendibile in Matlab)
Verifica Reale in z0= 3+1i : Non Olomorfa (Attendibile in Matlab)

ES. 20 – Quiz – Cauchy-Riemann

Quiz: perché la visualizzazione numerica locale delle Equazioni di Cauchy-Riemann presenta talvolta un taglio? Come può essere eliminato?

Risposta: Per alcune Funzioni Complesse, le Equazioni di Cauchy – Riemann presentano un Taglio, ossia una parte vuota visibile in ambito numerico ma non in ambiente simbolico.

Ciò accade perché in corrispondenza dei vuoti, l'array CR, contenente i Moduli della Matrice Complessa che calcola le Equazioni di Cauchy – Riemann, presenta dei valori NaN, dovuti ad alcuni risultati sul Calcolo del Limite del Rapporto Incrementale di CR. Tali valori rappresentano in pratica delle forme indeterminate del tipo 0/0, dovute al fatto che Matlab calcola separatamente numeratore e denominatore e poi effettua il rapporto.

Si è scelto di Eliminare questo taglio trovando nell'Array Tutti i Valori NaN e sostituirli con i Valori Complessi immediatamente prossimi ad essi in posizione all'interno di CR. Un Controllo IF, nel caso di Selezione della Modalità "NO Taglio", attiva l'Algoritmo di Correzione: quest'ultimo lavora sull'ipotesi che i NaN in CR sono sempre disposi a Forma di T

ES. n=4

NaN	**NaN**	**NaN**	**NaN**	**NaN**	**NaN**	**NaN**
0	1	2	**NaN**	4	5	6
7	8	9	**NaN**	10	11	12
13	14	15	**NaN**	16	17	18

In tale ipotesi per la Correzione sono stati elaborati

- 2 DOPPI CICLI FOR INNESTATI: Uno per sostituire i NaN della Parte sinistra, l'altro per quelli a Destra della Colonna Centrale
- 1 CICLO FOR: Per la sostituizione dei NaN della Colonna Centrale

NOTA1: Il costo dei cicli for è dispendioso se si creano array grandi

SOLUZIONE ALTERNATIVA : Un'alternativa più semplice sarebbe quella di usare un'unica Riga di Istruzione Correttiva: poiché per definizione, i NaN sono VALORI NON UGUALI A LORO STESSI

CR(CR~=CR)=rand(1);

Il Risultato della Disuguaglianza è una Matrice di Valori Booleani dello stesso Size di RC, e tali

Valori di Verità servono a Pilotare sulla stessa Matrice CR l'Assegnazione di un valore Reale Random ai soli elementi NaN

NOTA2: per trovare i valori NaN viene usata la funzione Matlab isNaN ( )

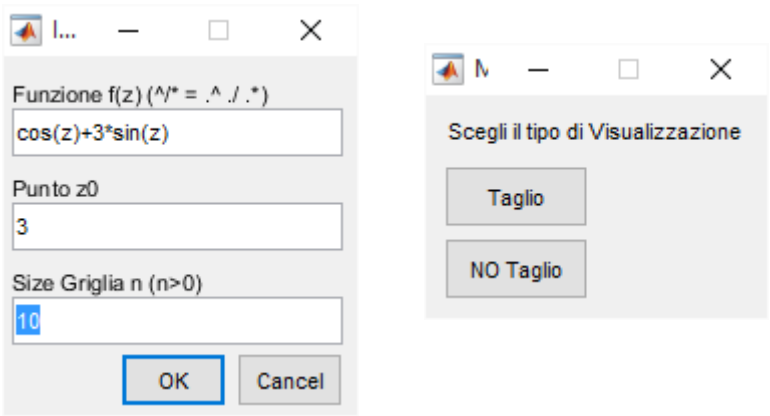
```
%   Matematica Applica e Computazionale
%
%   QUIZ CAUCHY - REIMANN - Liv 1
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
%
% USO
%
% Lanciare da consolle il file
%
% >> quiz_cauchyriemann
%
% e vedere i calcoli che esegue

%%%%%%%
% MAIN
%%%%%%%

% TRACCIA QUIZ %
% QUIZ: perché la visualizzazione numerica locale delle Equazioni di
% Cauchy-Riemann presenta talvolta un taglio? Come può essere eliminato?
uscita='No';

while strcmp(uscita,'No')

      % COMPONENTI DI INTERFACCIA GRAFICA UTENTE (GUI) PER L'ASSUNZIONE
DI INPUT
      campi={'Funzione f(z) (^/* = .^ ./ .*)','Punto z0','Size Griglia n
(n>0)'};
      titoloDLG='Input di Dati';
      numeroRighe=1;
```

```
valoriDefault={'z','0.1','10'};
  datiInput = inputdlg(campi,titoloDLG,numeroRighe,valoriDefault);
  datiInput = char(datiInput);
  f = datiInput(1,:);
  z0 = str2num(datiInput(2,:));
  n = str2num(datiInput(3,:));

  % CALCOLO NUMERICO - VERIFICA LOCALE NUMERICA DELLE EQUAZIONI DI
  % CAUCHY-RIEMANN
  fz=inline(f);
  Dz=cplxgrid(n); % griglia di n punti che discretizza Dz
  Dx=real(Dz);
  Dy=imag(Dz);
  x0=real(z0);
  y0=imag(z0);

  % verifica numerica locale delle equazioni di cauchy-riemann
  % CR: Cauchy Riemann
  CR=abs((fz(z0+Dx)-fz(z0))./Dx+1i*(fz(z0+1i*Dy)-fz(z0))./Dy);
  sceltaVisualizzazione=menu('Scegli il tipo di Visualizzazione',...
  'Taglio','NO Taglio');

  if sceltaVisualizzazione==2
        % --- VISUALIZZAZIONI SENZA TAGLIO ---
        %{
        Questo simbolo 祇ipo /* ... */ del Linguaggio C/C++
        PER OPERARE UNA CORREZIONE DEL TAGLIO DOBBIAMO TENER D'OCCHIO
        LA MATRICE CR, DATO CHE IL TAGLIO SI PRESENTA PERCHE' AL SUO
        INTERNO SONO PRESENTI VALORI NaN CHE VANNO SOSTITUITI. SE
        OSSERVIAMO BENE CI ACCORGIAMO CHE PER QUALSIASI FUNZIONE,
        TRIGONOMETRICA O NON, LA DISPOSIZIONE DI TALI NaN E' SEMPRE A
        FORMA DI CROCE O T

        ES. n=4
        NaN NaN NaN NaN NaN NaN NaN
         0   1   2  NaN  4   5   6
         7   8   9  NaN 10  11  12
        13  14  15  NaN 16  17  18

        QUINDI E' SUGGERIBILE SVILUPPARE UN ALGORITMO CHE SOSTITUISCA
        PRIMA I NaN DELLA PARTE DESTRA, POI QUELLI DELLA PARTE
        SINISTRA E INFINE QUELLI DELLA COLONNA CENTRALE.
```

```
        OPERO TUTTO CIO' CON UN DOPPIO CICLO FOR PER OGNI SEZIONE PER
        POTER SOSTITUIRE I NaN CON I VALORI REALI DI POSIZIONE PIU
        VICINA AD ESSI NELL'ARRAY 2D CR
        ------ ATTENZIONE!!! COSTO COMPUTAZIONALE ELEVATO! ------
        ALTERNATIVA:
        AVREMMO POTUTO USARE UN'ISTRUZIONE SIMILE ALLA SEGUENTE,
        SULLA BASE DELLA DEFINIZIONE DI VALORE NaN, ASSEGNANDO AD
        ESSI DEI REALI RANDOM  CR(CR~=CR)=rand(1); %NaN:valori
        IN MODO DA ANNULLARE I COSTI DEI CICLI FOR derivanti da forme
        indeterminate

        %}

        % CICLI FOR INNESTATI: SOSTITUISCE NaN DELLA PARTE SINISTRA
        for kk=1:n
              for ii=1:n
                    if isnan(CR(ii,kk))
                          CR(ii,kk)=CR(ii+1,kk);
                    end
              end
        end

        % CICLI FOR INNESTATI: SOSTITUISCE NaN DELLA PARTE DESTRA
        for kk=n+1:2*n+1
              for ii=1:n
                    if isnan(CR(ii,kk))
                          CR(ii,kk)=CR(ii+1,kk);
                    end
              end
        end

        % CICLO FOR SU COLONNA: SOSTITUISCE NaN DELLA COLONNA
        % CENTRALE
        kk=n+1;

        for ii=1:n+1
              if isnan(CR(ii,kk))
                    CR(ii,kk)=CR(ii,kk-1);
              end
        end

end % END IF
```

```
        % - VISUALIZZAZIONI CON TAGLIO (SE NON SI E' ENTRATI NELL'IF) -%
        % LE SEGUENTI VISUALIZZAZIONI PRESENTANO UN TAGLIO PER ALCUNE
        % FUNZIONI PERCHE' ALCUNI PUNTI DELLA MATRICE DI CAUCHY RIEMANN
        % POSSONO RISULTARE NaN. SE SI E' ENTRATI NELL'IF PER ELIMINARE
        % TALE TAGLIO IL BUCO DELL'INDETERMINAZIONE VERRA' COPERTO CON UN
        % COMPLESSO RANDOM
        % SURFACE surf()
        % DISEGNO DELLA FUNZIONE IN FORMA COMPLESSA DELLE EQUAZIONI
        % DI EQUAZIONI DI C.R.
        figure
        surf(x0+Dx, y0+Dy, CR);

        % COMPLEX MAP cplxmap()
        % DISEGNO DELLA FUNZIONE IN FORMA COMPLESSA DELLE EQUAZIONI DI C.R.
        figure
        cplxmap(z0+Dz,CR*(1+1i));

        % CRITERI DI SELEZIONE USCITA PER GESTIRE CICLO WHILE
        % ATTRAVERSO UNA QUESTION DIALOG BOX
        uscita=questdlg('Vuoi uscire?','Continua...');

end % END WHILE
```

Esempio d’uso



Secgliendo **Taglio**, otteniamo

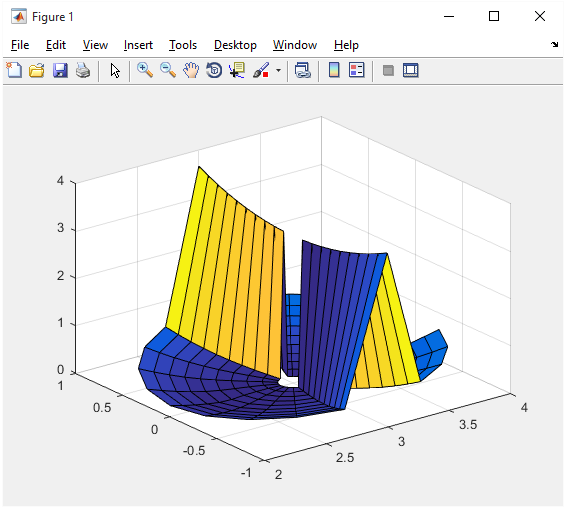


Figure 1
File Edit View Insert Tools Desktop Window Help

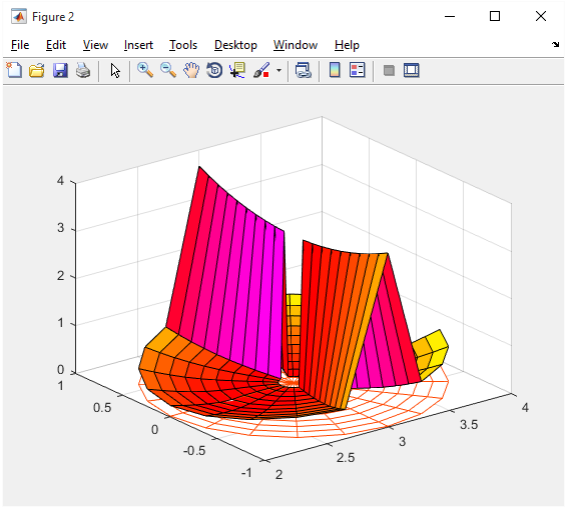


Figure 2
File Edit View Insert Tools Desktop Window Help

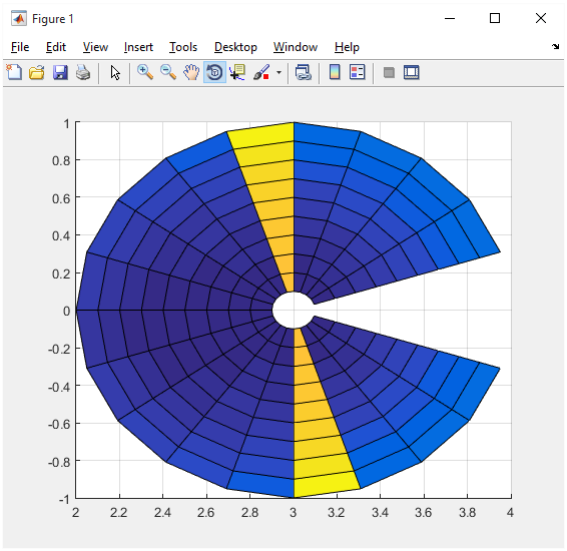


Figure 1
File Edit View Insert Tools Desktop Window Help

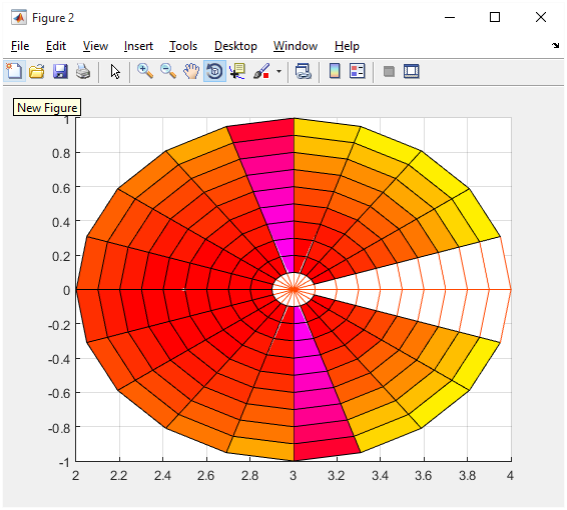
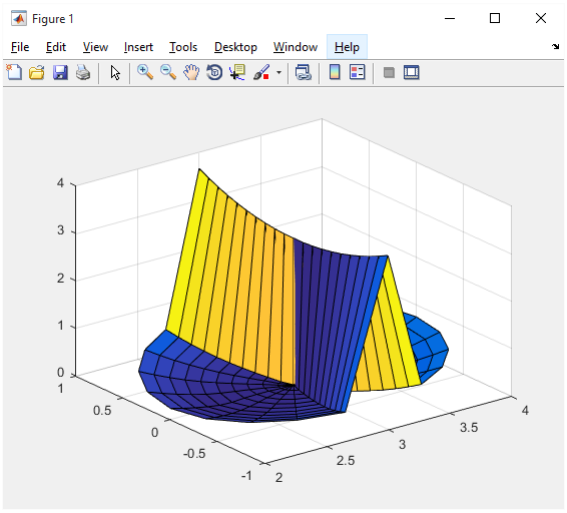
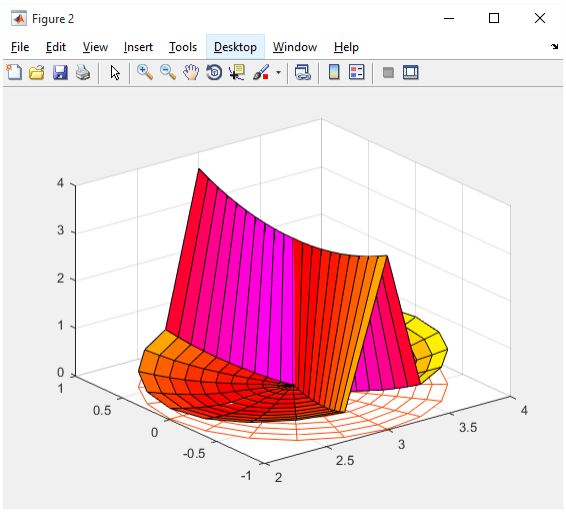


Figure 2
File Edit View Insert Tools Desktop Window Help
New Figure

Se invece scegliamo **No Taglio**, otteniamo






```
CR =

  Columns 1 through 11

       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN
    0.0231    0.0234    0.0243    0.0256    0.0272    3.1164    0.0304    0.0317    0.0327    0.0333       NaN
    0.0358    0.0374    0.0417    0.0476    0.0542    3.1324    0.0662    0.0709    0.0744    0.0765       NaN
    0.0379    0.0428    0.0546    0.0692    0.0839    3.1591    0.1094    0.1187    0.1255    0.1295       NaN
    0.0295    0.0425    0.0674    0.0940    0.1194    3.1968    0.1610    0.1758    0.1861    0.1921       NaN
    0.0107    0.0440    0.0854    0.1255    0.1627    3.2456    0.2221    0.2424    0.2563    0.2642       NaN
    0.0184    0.0590    0.1130    0.1660    0.2153    3.3058    0.2933    0.3191    0.3360    0.3453       NaN
    0.0576    0.0912    0.1523    0.2169    0.2785    3.3778    0.3753    0.4059    0.4251    0.4352       NaN
    0.1066    0.1374    0.2034    0.2790    0.3529    3.4620    0.4685    0.5032    0.5236    0.5334       NaN
    0.1651    0.1953    0.2662    0.3525    0.4392    3.5589    0.5734    0.6111    0.6311    0.6395       NaN
    0.2327    0.2635    0.3401    0.4379    0.5382    3.6691    0.6908    0.7298    0.7476    0.7529       NaN

  Columns 12 through 21

       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN
    0.0333    0.0327    0.0317    0.0304    3.1164    0.0272    0.0256    0.0243    0.0234    0.0231
    0.0765    0.0744    0.0709    0.0662    3.1324    0.0542    0.0476    0.0417    0.0374    0.0358
    0.1295    0.1255    0.1187    0.1094    3.1591    0.0839    0.0692    0.0546    0.0428    0.0379
    0.1921    0.1861    0.1758    0.1610    3.1968    0.1194    0.0940    0.0674    0.0425    0.0295
    0.2642    0.2563    0.2424    0.2221    3.2456    0.1627    0.1255    0.0854    0.0440    0.0107
    0.3453    0.3360    0.3191    0.2933    3.3058    0.2153    0.1660    0.1130    0.0590    0.0184
    0.4352    0.4251    0.4059    0.3753    3.3778    0.2785    0.2169    0.1523    0.0912    0.0576
    0.5334    0.5236    0.5032    0.4685    3.4620    0.3529    0.2790    0.2034    0.1374    0.1066
    0.6395    0.6311    0.6111    0.5734    3.5589    0.4392    0.3525    0.2662    0.1953    0.1651
    0.7529    0.7476    0.7298    0.6908    3.6691    0.5382    0.4379    0.3401    0.2635    0.2327
```

ES. 21 – Liv. 2 – Quiz Cauchy-Riemann

Verificare graficamente in una regione del piano complesso le Equazioni di Cauchy-Riemann per le funzioni (§) mediante la function `gradient()` di MATLAB. Che differenza c'è nell'usare `jacobian()` al posto di `gradient()`?

Quiz: Perché la visualizzazione dell'output di `gradient()` ha una forma "a cassetto"?

Soluzione proposta

Il programma presenta l'Algoritmo di Valutazione dell'Olomorfia di una Funzione Complessa utilizzando la MATLAB Function `gradient()`.

Tale funzione, a differenza dell'Algoritmo con la Matrice Jacobiana (usato nell'esercizio 18), restituisce il gradiente di f, ovvero il vettore contenente i rapporti Df/Dx e Df/Dy (derivate parziali di f) per la Forma Complessa dell'Olomorfia (Poco Attendibile).

L'Algoritmo che sfrutta jacobian(), utilizza il gradiente per una Verifica dell'Olomorfia in Forma Reale (Sicuramente Attendibile), prendendo in input

Du/Dx (real(f)/real(z0))

Dv/Dy (imag(f)/imag(z0))

Altra differenza è che nell'Algoritmo con gradient() non si riscontra MAI IL PROBLEMA DEL TAGLIO, che alle volte si presenta invece con l'Algoritmo Jacobiano

Infine, `gradient()`, ha un Metodo di Visualizzazione dell'Olomorfia Differente dall'Algoritmo Jacobian, ovvero presenta quantomeno sempre una Curva che ha una Parte Piana posta a Quota Spaziale 0 nell'Area dove che indica i Punti dove la Funzione è Olomorfa e in casi particolari, tale Curva assume proprio una Forma Piana con Bordo come un Cassetto.

```
%   Matematica Applicata e Computazionale
%
%   Cauchy - Riemann Gradient - Liv 2
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
%
% USO
%
% Lanciare da consolle il file
%
% >> CRgradient
%
% e vedere i calcoli che esegue

%%%%%%%
% MAIN
%%%%%%%

uscita='No';
f_CRgradient(uscita);
```

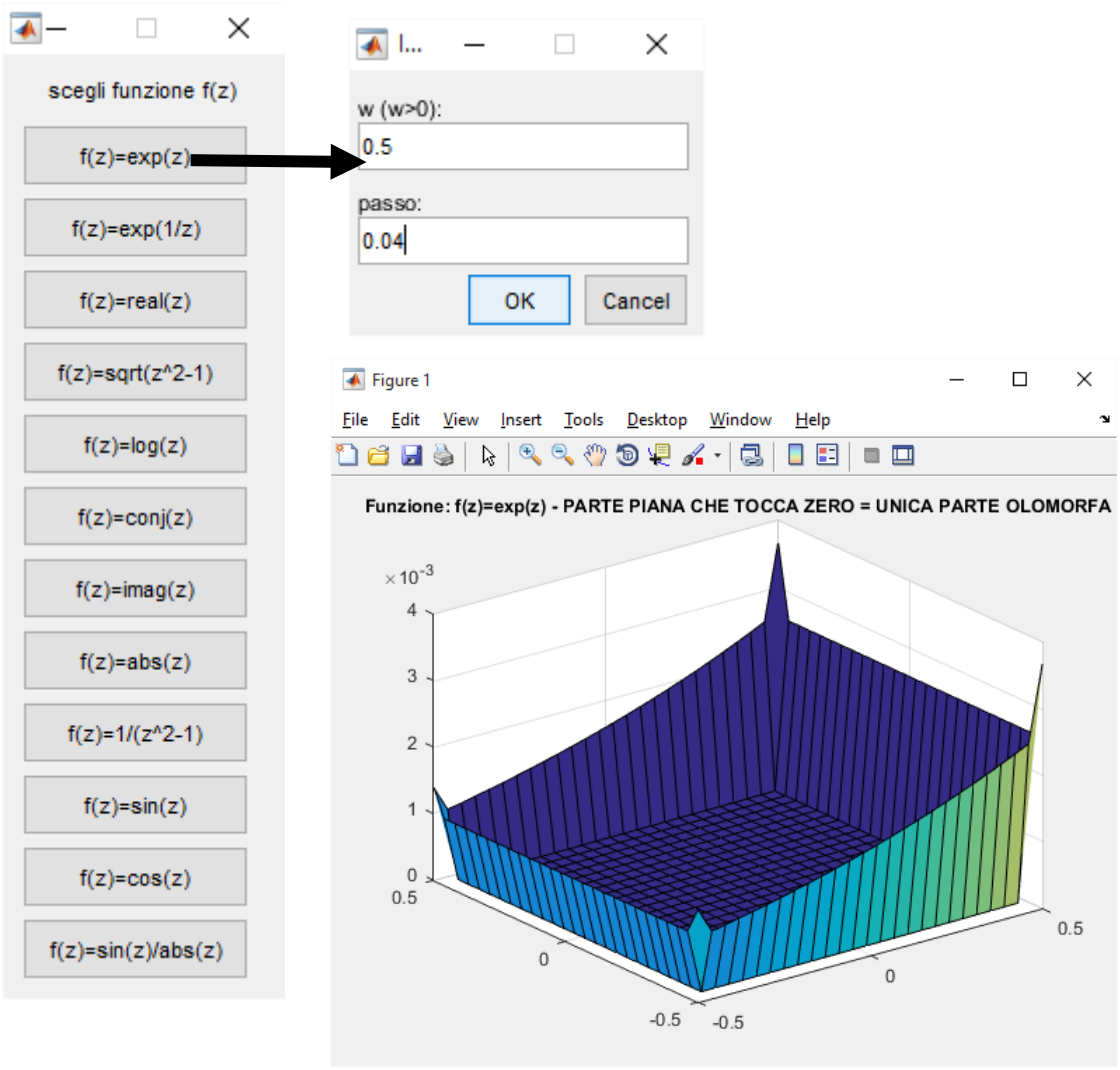
```
%   Matematica Applicata e Computazionale
%
%   Cauchy - Riemann Gradient - Liv 2
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%


%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function f_CRgradient.m
%%%%%%%%%%%%%%%%%%%%%%%%%%

% QUIZ: Perchè la Visualizzazione dell'output di gradient() ha una forma
a cassetto?

% LA FUNZIONE gradient() RESTITUISCE UN GRAFICO DALLA FORMA A CASSETTO,
% O QUANTOMENO, UNA FORMA IRREGOLARE CHE PRESENTA UNA PARTE PIANA.
% IN ENTRAMBI I CASI, SIA LA PARTE PIANA CHE LA BASE DEL CASSETTO SI
% TROVANO AD UNA ALTEZZA 0 RISPETTO L'ASSE z CIO' INDICA CHE IN QUEI
% PUNTI LA FUNZIONE E' OLOMORFA POICHE' L'EQUAZIONE
% IN FORMA COMPLESSA CR=DfDx+1i*DfDy ASSUME IVI VALORE 0
% LE PARTI RIMANENTI DEI GRAFICI, NON HANNO PUNTI CHE TOCCANO TALE VALORE
% QUINDI INDICANO DOVE LA FUNZIONE NON E' OLOMORFA O DOVE COMINCIA AD
% ALLONTANARSI DALLA CONDIZIONE DI OLOMORFIA

function f_CRgradient(uscita)
while strcmp(uscita,'No')
      % RICHIAMA LA FUNCTION PER LA SELEZIONE DELLA FUNZIONE DA DISEGNARE
      [funzione, fStringa] = scelta_funzione();
      campi={'w (w>0): ','passo:'};
      titoloDLG='Input di Dati';
      numeroRighe=1;
      valoriDefault={'1','0.1'};
```

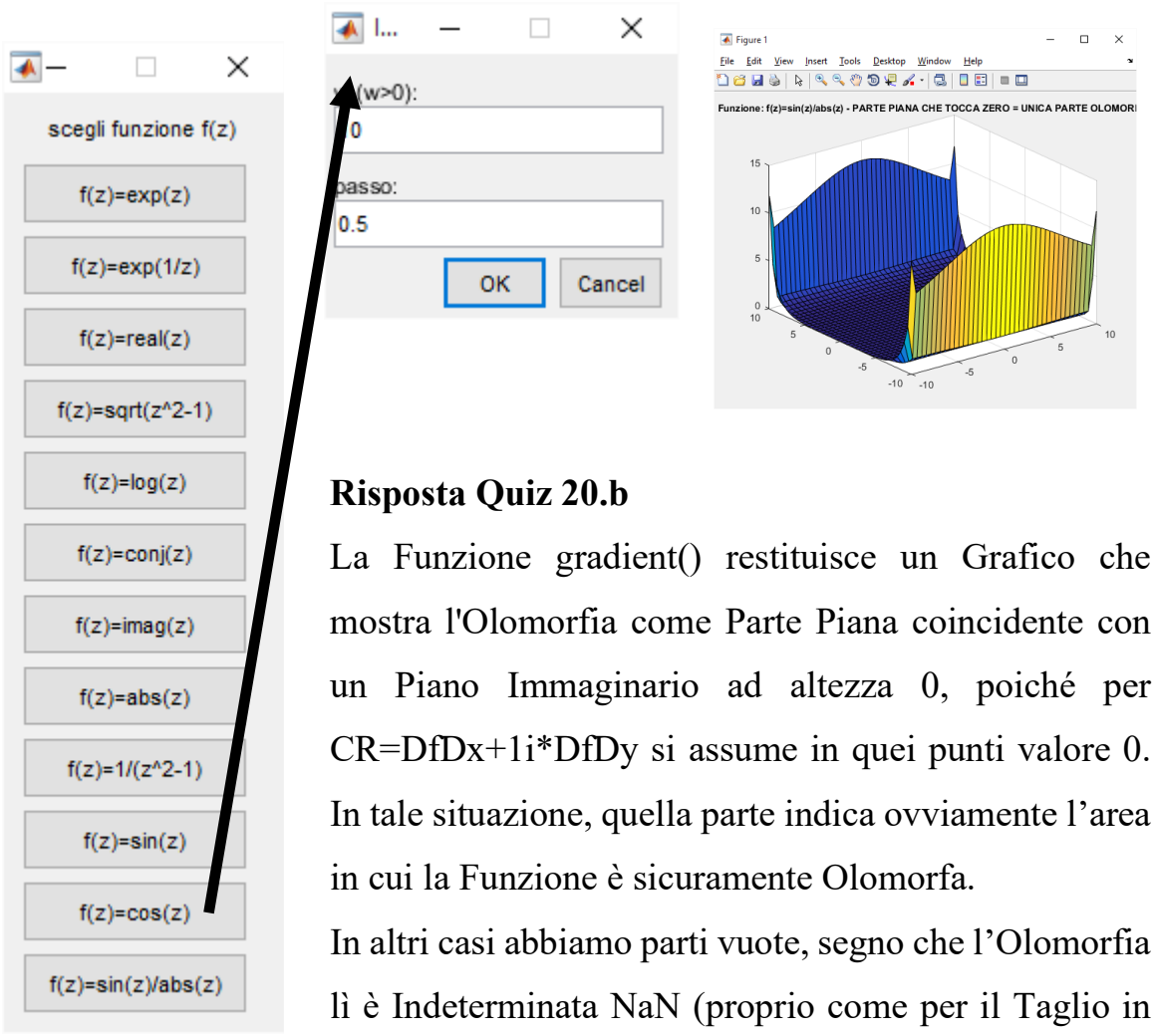
```
        datiInput = inputdlg(campi,titoloDLG,numeroRighe,valoriDefault);
        datiInput = char(datiInput);
        datiInput = str2num(datiInput);
        w = datiInput(1);
        passo = datiInput(2); % serve a disegnare la girglia con meshgrid
        [x,y]=meshgrid(-w:passo:w);
        z=x+1i*y;
        fz=funzione(z); % VAOLRE DELLA FUNZOIONE IN z
        [DfDx,DfDy]=gradient(fz,w); % gradiente della funzione
        CR=DfDx+1i*DfDy; % verifica equaioni di Cauchy Riemann

        figure
        surf(x,y,abs(CR))
        title(['Funzione: ' fStringa ' - PARTE PIANA CHE TOCCA '...
        'ZERO = UNICA PARTE OLOMORFA' ],'FontSize',10);

        % CRITERI DI SELEZIONE USCITA PER GESTIRE CICLO WHILE
        % ATTRAVERSO UNA QUESTION DIALOG BOX
        uscita=questdlg('Vuoi uscire?','Continua...');
end % END WHILE

end % END FUNCTION
```

```
%
%   Matematica Applicata e Computazionale
%
%   Cauchy - Riemann Gradient - Liv 2
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function: scelta_funzione.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [f, fString] = scelta_funzione()

% FUNZIONI TEST
cellFunzioni={'f(z)=exp(z)'; 'f(z)=exp(1/z)'; 'f(z)=real(z)';
'f(z)=sqrt(z^2-1)';'f(z)=log(z)';'f(z)=conj(z)';
'f(z)=imag(z)';'f(z)=abs(z)';'f(z)=1/(z^2-1)';
'f(z)=sin(z)';'f(z)=cos(z)';'f(z)=sin(z)/abs(z)';};

funzioneScelta = menu('scegli funzione f(z)',cellFunzioni);

switch funzioneScelta
      case 1
            f=@(z) exp(z);

      case 2
            f=@(z) exp(1./z);

      case 3
            f=@(z) real(z);
```

```
        case 4
              f=@(z) sqrt((z.^2)-1);

        case 5
              f=@(z) log(z);

        case 6
              f=@(z) conj(z);

        case 7
              f=@(z) imag(z);

        case 8
              f=@(z) abs(z);

        case 9
              f=@(z) 1/((z.^2)-1);

        case 10
              f=@(z) sin(z);

        case 11
              f=@(z) cos(z);

        case 12
              f=@(z) sin(z)./abs(z);

        end % END SWITCH

        fString=cellFunzioni{funzioneScelta};

end % END Function
```

Esempio d'uso





Risposta Quiz 20.b

La Funzione gradient() restituisce un Grafico che mostra l'Olomorfia come Parte Piana coincidente con un Piano Immaginario ad altezza 0, poiché per CR=DfDx+1i*DfDy si assume in quei punti valore 0. In tale situazione, quella parte indica ovviamente l'area in cui la Funzione è sicuramente Olomorfa.

In altri casi abbiamo parti vuote, segno che l'Olomorfia lì è Indeterminata NaN (proprio come per il Taglio in Jacobian).

Infine abbiamo un ultimo caso particolare: sembra che in determinate circostanze, alcuni Grafici di Funzioni Non Completamente Olomorfe, presentino delle parti in cui la Curva all'Improvviso Sale Vorticosamente, rendendo le sembianze del Grafico simili ad una Forma a Cassetto. Ciò avviene perché in quei punti la Curva avrà evidentemente Valori Lontani da 0 (tale valore rammentiamo che indica Olomorfia) ; di Conseguenza i "Bordi di Tale Cassetto" o comunque ogni irregolarità della Curva che mostri sollevamenti o Allontanamenti dallo 0, indicano che in tale zona della Curva la Funzione Non E' Olomorfa

Sucessioni e Serie

Es. 22– Quiz dei cubi

Quiz: Supponendo di avere infiniti cubi di lato $1/n$ (in metri) $\forall n \in \mathrm{N}$:

1. Che altezza raggiungerà la torre con gli infiniti cubi sovrapposti uno sull'altro?
2. Quale superficie sarà richiesta per contenere tutti i cubi affiancati?
3. Quale volume sarà complessivamente occupato dai cubi?

Soluzione proposta

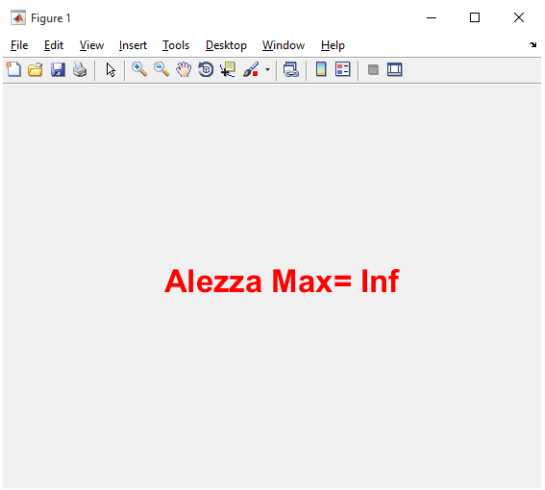
La risoluzione di questo quiz prevede l'applicazione della serie armonica (per il calcolo
dell'altezza), della serie armonica generalizzata di esponente 2 (per il calcolo dell'area) e di esponente 3 (per il calcolo dell'area).
Il codice Matlab consta di un unico script dimostrativo

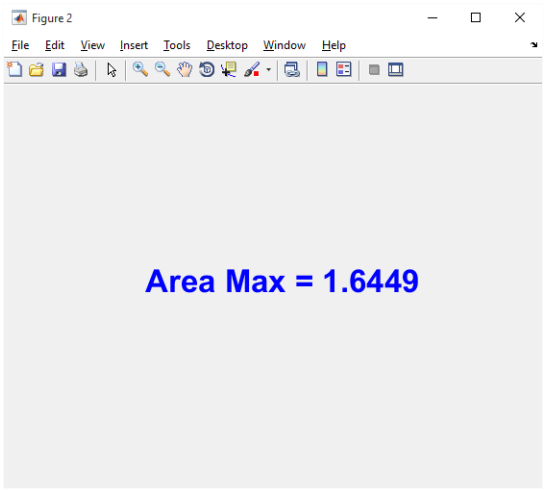
```
%   Matematica Applica e Computazionale
%
%  Quiz Cubi
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
%
% USO
%
% Lanciare da consolle il file
%
% >> quizcubi
%
% e vedere i calcoli che esegue

% QUIZ: Supponendo di avere infiniti cubi di lato 1/n (in metri) ?n?N:
% 1) Che altezza raggiungerà la torre con gli infiniti cubi sovrapposti
% uno sull'altro?
% 2) Quale superficie sarà richiesta per contenere tutti i cubi affiancati?
% 3) Quale volume sarà complessivamente occupato dai cubi?

syms n;
format rat
lato=1/n; % LATO DI UN CUBO

% L'ALTEZZA TOTALE E' CALCOLATA COME SOMMATORIA SIMBOLICA symsum()
% DEGLI LATI DEGLI n CUBI PER n CHE VA DA 0 A INFINITO
display('Altezza Massima');
altezzaTot=symsum(lato,n,1,Inf) %SOMMATORIA
area=(1/n)^2; % AREA DI UN CUBO
```

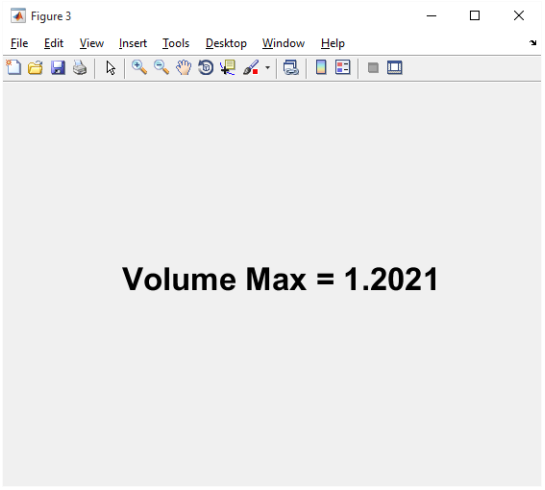
```
% L'AREA TOTALE E' CALCOLATA COME SOMMATORIA SIMBOLICA symsum()
% DELLE BASI DEGLI n CUBI PER n CHE VA DA 0 A INFINITO
display('Area Massima');
areaTot=symsum(area,n,1,Inf)
volume=1/(n^3); % VOLUME DI UN CUBO

% IL VOLUME TOTALE E' CALCOLATO COME SOMMATORIA SIMBOLICA symsum()
% DEI VOLUMI DEGLI n CUBI PER n CHE VA DA 0 A INFINITO
display('Volume Massimo')
volumeTot=symsum(volume,n,1,Inf)

% SE NON APPLICHIAMO double() RESTITUISCE LA FUNZIONE SYMBOLICA
% zeta() DI RIEMANN
strAlt=['Alezza Max= ' num2str(double(altezzaTot))];
strArea=['Area Max = ' num2str(double(areaTot))];
strVol=['Volume Max = ' num2str(double(volumeTot))];
text(.5,.5,strAlt,'FontSize',25,'HorizontalAlignment','center',...
'Color','r','FontWeight','bold');
axis off
figure

text(.5,.5,strArea,'FontSize',25,'HorizontalAlignment','center',...
'Color','b','FontWeight','bold');
axis off
figure

text(.5,.5,strVol,'FontSize',25,'HorizontalAlignment','center',...
'Color','k','FontWeight','bold');
axis off
```

Esempio d'uso

```
>> quizcubi
Altezza Massima

altezzaTot =

Inf

Area Massima

areaTot =

pi^2/6

Volume Massimo

volumeTot =

zeta(3)
```







Es. 24– Liv. 2 – Serie numeriche

Mediante *function MATLAB* approssimare numericamente a *p* cifre decimali corrette la somma delle seguenti serie numeriche:

$$1 + \frac{1}{2!} + \frac{1}{3!} + \cdots$$

$$1 + \frac{1}{2^2} + \frac{1}{3^2} + \cdots$$

$$1 + \frac{1}{2^4} + \frac{1}{3^4} + \cdots$$

[*Suggerimento*: funzioni quad() o quadl() o quadgk() per approssimare numericamente un integrale.]

Soluzione Proposta

Questo elaborato seleziona da Menu Grafico una delle 3 Serie Numeriche contemplate nel testo, e prende in input il Numero di Cifre da approssimare. Sostanzialmente il Programma è composto da 3 blocchi

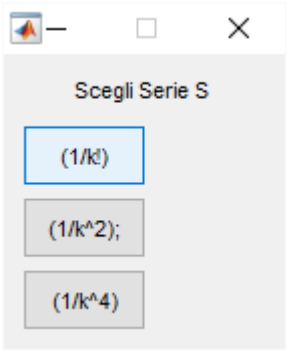
1. SCRIPT:(sym_serie) Ha funzioni di Input attraverso Interfacce Utente e visualizza il risultato in una Finestra text()
2. FUNCTION: (f_symserie, scelta_serie) La prima ha funzioni di Calcolo Matematico, la seconda Crea e Restituisce la Serie Numerica scelta.
   La f_symserie sviluppa l'Algoritmo di Calcolo sulla Somma S della Serie e sulla sua Ridotta Sn di Ordine ind generico, che parte da 1, per una Ridotta di Ordine Primo, fino ad Arrivare al Grado Massimo per il quale il Resto Rn avuto dalla Differenza fra la Sommatoria Completa S e la Ridotta di Ordine ind, ha Valore uguale al Numero di Cifre da Approssimare. Tutto ciò è gestito da un Ciclo WHILE, che incrementa ad ogni iterazione l'Ordine di Ridotta e ricalcola la Ridotta, finché il Resto non è Rn > 10p

```
%
%   Matematica Applica e Computazionale
%
%   Serie Numeriche
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                Â© 2016
%
%  GNU/GPL License
%
%
%
% USO
%
% Lanciare da consolle il file
%
% >> symserie
%
% e vedere i calcoli che esegue

%%%%%%%
% Main
%%%%%%%

clc

uscita='No';
syms k
while strcmp(uscita,'No')

    % MENU DI SCELTA DELLA SERIE
    % chiamata alla funzione che sceglie la serie su cui lavorare
    % ASSEGNA AD ak IL VALORE DEL TERMINE DELLA SERIE NUMERICA
    % ed a serieString UNA STRINGA COL NOME DELLA SOMMA DELLA SERIE
scelta
    [ak, serieString] = scelta_serie();

    % MENU' DI SCELTA DEL NUMERO DI CIFRE DA APPROSSIMARE
    campi={'Num. Approssimazione Cifre (p>0))'};
    titoloDLG='Input di Dati';
    numeroRighe=1;
    valoriDefault={'1'};
    datiInput = inputdlg(campi,titoloDLG,numeroRighe,valoriDefault);
    datiInput = char(datiInput);

    %%%%%%%%%%%%%%%%%%%%%%%% ATTENZIONE!!!%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % IL PROGRAMMA NON RIESCE AD ASSUMERE IL VALORE p=10 O SUPERIORE
    % PER FINITEZZA DELLA PRECISIONE IN MATLAB. RIESCE QUINDI AD
    % APPROSSIMARE SOLO UN NUMERO DI CIFRE INFERIORE A 10 p<10
    % ORDINE DELL'APPROSSIMAZIONE - NUMERO CIFRE
    p = str2num(datiInput(1));

    % CHIAMATA A FUNCTION PER CALCOLO del resto Rn
    [Rn]=fsym_approssCifre(ak,k,p);
```

```
    % SET DELL'OUTPUT A FINESTRA GRAFICA
    textOutput=['La Serie ' serieString ' Approssimata a '...
    num2str(p) ' Cifre come S=' num2str(Rn) ];
    figure
    text(.5,.5,textOutput,'FontSize',12,'HorizontalAlignment',...
    'center','Color','b','FontWeight','bold');
    axis off

    % CRITERI DI SELEZIONE USCITA DA WHILE TRAMITE QUESTION DIALOG BOX
    uscita=questdlg('Vuoi uscire?','Continua...');

end % END WHILE
```

```
%
%   Matematica Applica e Computazionale
%
%   Serie Numeriche
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                Â© 2016
%
%  GNU/GPL License
%
%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function fsym_approssCifre.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Rn = fsym_approssCifre(ak,k,p)
ind=1;
S=symsum(ak,k,1,inf); % CALCOLO DELLA SOMMA DELLA SERIE
Sn=symsum(ak,k,1,ind); % CALCOLO RIDOTTA DELLA SERIE DI ORDINE ind=1
Rn=subs(S-Sn); %CALCOLO DEL RESTO DELLA SERIE COME DIFFERENZA

% CICLO WHILE
% FINTANTO CHE IL RESTO E' MAGGIORE DELL'APPROSSIMAZIONE RICHIESTA
while (Rn>=10^-p)
    ind=ind+1; % INCREMENTA L'ORDINE DELLA RIDOTTA
    Sn=symsum(ak,k,1,ind); % CALCOLA RIDOTTA DELLA NUOVO ORDINE DELLA
SERIE
    Rn=subs(S-Sn); % CALCOLA RESTO DALLA RIDOTTA DEL NUOVO ORDINE
end

end % END FUNCTION
```

```
%
%   Matematica Applica e Computazionale
%
%   Serie Numeriche
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                Â© 2016
%
%  GNU/GPL License
%


%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function scelta_serie.m
%%%%%%%%%%%%%%%%%%%%%%%%%%

%GENERA UN MENU' DI SERIE DA SCEGLIERE E RESTITUISCE SIA
%UNA FUNZIONE RAPPRESENTATE IL TERMINE GENERICO DELLA SERIE SIMBOLICA
%SIA UNA STRINGA CHE RAPPRESENTA IL NOME DELLA SERIE
function [ak,serieString] = scelta_serie()

syms k;

% CELL ARRAY DI STRINGHE PER TITOLI BOTTONI DEL MENU' FUNZIONE
cellSerie={'(1/k!)';'(1/k^2);';'(1/k^4)'};

% MENU DI SCELTA DELLA SERIE
serieScelta = menu('Scegli Serie S',cellSerie);

% A SECONDA DELLA SERIE SCELTA ASSEGNO AD ak IL VALORE DEL
% GENERICO TERMINE DELLA SERIE
switch serieScelta
    case 1
        % TERMINE ak DELLA SERIE 1/k!
        ak=1/gamma(k+1);
    case 2
        % TERMINE ak DELLA SERIE 1/k^2
        ak=1/k^2;
    case 3
        % TERMINE ak DELLA SERIE 1/k^4
        ak=1/k^4;
end % END SWITCH

% ASSEGNA A serieString IL NOME DELLA SERIE SCELTA INDICIZZANDO
% CORRETTAMENTE IL CELL ARRAY DEFINITO cellSerie
serieString=cellSerie{serieScelta};

end % END FUNCTION
```


Scegli Serie S
(1/k!)
(1/k^2);
(1/k^4)

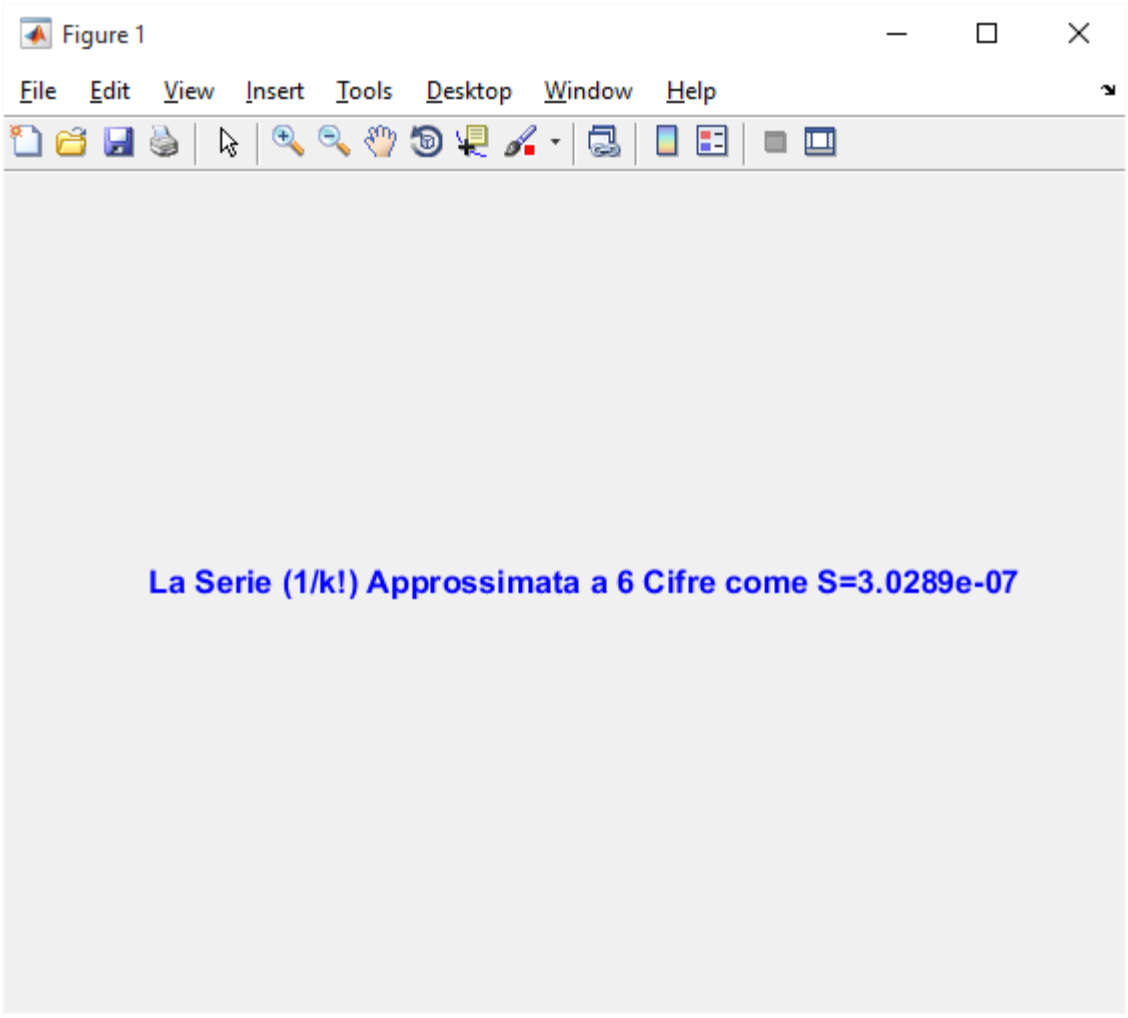
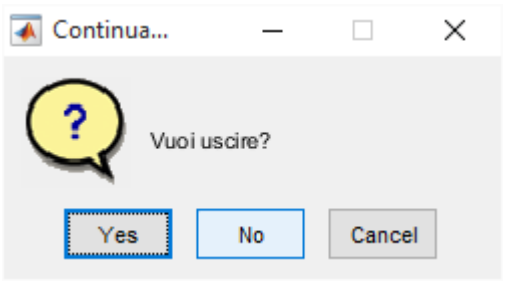


Figure 1
File Edit View Insert Tools Desktop Window Help
La Serie (1/k!) Approssimata a 6 Cifre come S=3.0289e-07


Continua...
Vuoi uscire?
Yes
No
Cancel

Es. 25– Liv. 2 – Studia Serie

Mediante il Symbolic Math Toolbox di MATLAB studiare la serie reale (riconducendola ad una serie nota): $1-x^2+x^4-x^6+\ldots$

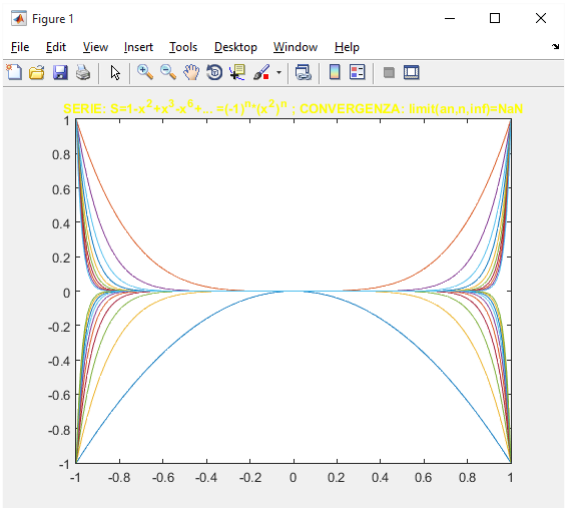
Soluzione Proposta

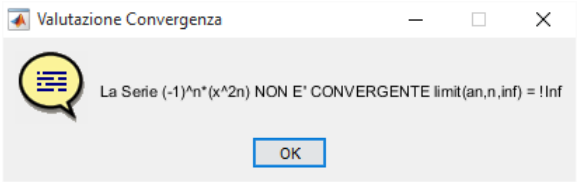
L'elaborato si compone di un unico script che analizza la serie data nel testo.
Tale serie è (-1)^n * x^2n
Rammentando LA PROPRIETA' DELLE POTENZE, useremo la notazione
x^2n = (x^2)^n)
per evitare un certo disordine della funzione text() che non gestisce (graficamente) bene il testo dell'elevazione a potenza nel prodotto coi fattori
x^2*n

```
%   Matematica Applicata e Computazionale
%
%   Serie Numeriche - Studio di Serie
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
%
% USO
%
% Lanciare da consolle il file
%
% >> studiaserie
%
% e vedere i calcoli che esegue

syms x real
syms n positive

% LA SERIE E' an=(-1^n)*(x^2n) TUTTAVIA PER LA PROPRIETA' DELLE POTENZE
% SI HA x^2n = (x^2)^n), USEREMO PERTANTO LA SECONDA MODALITA' DI
% NOTAZIONE PER EVITARE UN CERTO DISORDINE DELLA FUNZIONE text()
% CHE NON GESTISCE GRAFICAMENTE BENE IL TESTO DELL'ELEVAZIONE A POTENZA
% DI NEL PRODOTTO x^2*n

an=((-1)^n)*(x^2)^n;
titolo=['STUDIO SERIE [S=1 - x^2 + x^3 - x^6 + ...]'...
' ricondotta a Serie Nota S= ' char(an)];
figure
axis off

try % caso in cui la serie è convergente
      convergenza=limit(an,n,inf);
```

```
        convergenza=double(convergenza);
        helpdlg(['La Serie (-1)^n*(x^2n) e'' CONVERGENTE'...
        ' limit(an,n,inf) != Inf'],'Valutazione Convergenza');
catch
        % LA SERIE NOTA E' INDETERMINATA
        % LA SERIE E' A TERMINI DI SEGNO ALTERNATO PER L'APPLICAZIONE
        % DELLA POTENZA DI n SULLA BASE (-1), HA QUINDI POTENZE SI'
        % CRESCENTI MULTIPLE DI 2 MA DI SEGNO OPPOSTO FRA UN TERMINE ED IL
        % SUO SUCCESSIVO. DI CONSEGUENZA LA FUNZIONE E' INDETERMINATA
        convergenza=limit(an,n,inf);
        convergenza=subs(convergenza,{x},{1});
        helpdlg(['La Serie (-1)^n*(x^2n) NON E'' CONVERGENTE'...
        ' limit(an,n,inf) = !Inf'],'Valutazione Convergenza');

end % end try - catch

% Stampa grafico
[nn,xx]=meshgrid(1:20,linspace(-1,1,200));
Sn=((-1).^nn).*(xx.^(2*nn));
plot(xx(:,1),Sn);
titolo=['SERIE: S=1-x^2+x^3-x^6+... =' char(an)...
' ; CONVERGENZA: limit(an,n,inf)=' char(convergenza)];
title(titolo,'FontSize',10,'Color','y','FontWeight','bold');
```

Esempio d’uso





Serie di potenze nel campo del complesso

Es. 28– Li. 1 – Serie Potenza

Visualizzare la convergenza (in un punto) di serie note come somma di vettori applicati del piano.

Soluzione Proposta

L'Algebra Lineare consente di visualizzazione la Convergenza di una Serie di Potenze Complesse in un punto z, rappresentando i Termini Complessi come Vettori del piano R2 .

Il nostro Elaborato consente di scegliere da Menu Grafici i dati di input come la Serie di Potenze desiderata, che non viene mai costruita veramente, in quanto il Ciclo WHILE che gestisce la Costruzione Continua di ogni nuova Ridotta Successiva Sn di Nuovo Ordine k, effettua tale operazioni per Somme di Componenti Successive, e tali Componenti sono generate di volta in volta con una inline(). Il Disegno di Tali Vettori è ANIMATO grazie a pause(0.25), che rallenta l'esecuzione di ¼ di secondo per ogni iterazione

I restanti input sono la scelta del tipo di Coordinate del Punto di Convergenza z, con relativa gestione seguente attraverso un Costrutto SWITCH CASE, e l'Inserimento dei loro valori più quello dell'Ordine Massimo, attraverso una Input Dialog Box

Nota: PER TESTARE IL PROGRAMMA, AFFINCHE' OGNI VETTORE RIMPICCIOLISCA E NON AUMENTI DI VOLTA IN VOLTA, E' CONSIGLIABILE UTILIZZARE VALORI DI a E b, rho e theta PICCOLI, O QUANTOMENO INFERIORI ALL'UNITÀ ES: a=0.8620 b=0.5000
rho=1 theta = pi/6

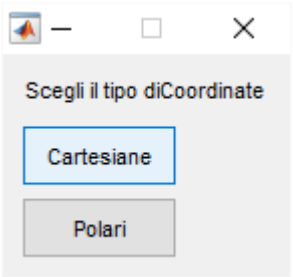
```
%   Matematica Applica e Computazionale
%
%   Serie di Potenze - Campo di Convergenza
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
%
% USO
%
% Lanciare da consolle il file
%
% >> seriepotenza
%
% e vedere i calcoli che esegue

%%%%%%%
% Main
%%%%%%%

clc % pulisce la schermata di matlab

uscita='No';
f_seriepotenze(uscita);
```

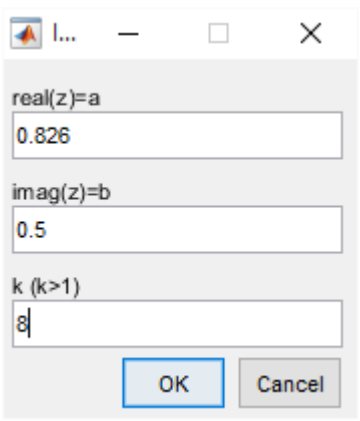
```
%   Matematica Applicata e Computazionale
%
%   Serie di Potenze - Campo di Convergenza
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%

%%%%%%%%%%%%%%%%%%%
% f_seriepotenze.m
%%%%%%%%%%%%%%%%%%%

function f_seriepotenze(uscita)

while strcmp(uscita,'No')
      % MENU' DI SCELTA FUNZIONE
                sceltaCoordinate = menu('Scegli il tipo di
                      Coordinate','Cartesiane','Polari');

    switch sceltaCoordinate
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            % *************** ATTENZIONE!! ******************
            % PER TESTARE IL PROGRAMMA, AFFINCHE' OGNI VETTORE
            % RIMPICCIOLISCA E NON AUMENTI DI VOLTA IN VOLTA
            % E' CONSIGLIABILE UTILIZZARE VALORI DI a E b PICCOLI,
            % O QUANTOMENO INFERIORI ALL'UNITA'
            % ES: a=0.8620 b=0.5000
            % rho=1 theta=pi/6
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            case 1
                  % MENU' DI SCELTA DELLE COMPONENTI CARTESIANE [a,b] E
DELL'ORDINE k
                  campi={'real(z)=a','imag(z)=b','k (k>1)'};
```

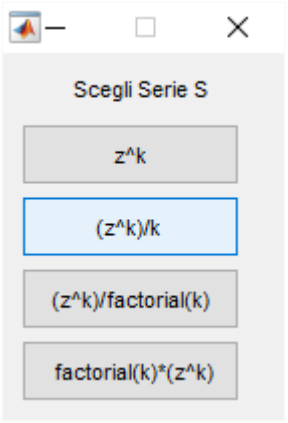
```
                titoloDLG='Input di Dati';
                numeroRighe=1;
                valoriDefault={'0.1','0.1','1'};
                datiInput                                                    =
inputdlg(campi,titoloDLG,numeroRighe,valoriDefault);
                datiInput = char(datiInput);
                datiInput = str2num(datiInput);
                a=datiInput(1);
                b=datiInput(2);
                z=a+1i*b;

            case 2
                % MENU' DI SCELTA DELLE COMPONENTI POLARI [rho,theta] E
DELL'ORDINE k
                campi={'abs(z)=rho','imag(z)=theta','k (k>1)'};
                titoloDLG='Input di Dati';
                numeroRighe=1;
                valoriDefault={'0.1','0.1','1'};
                datiInput                                                    =
inputdlg(campi,titoloDLG,numeroRighe,valoriDefault);
                datiInput = char(datiInput);
                datiInput = str2num(datiInput);
                rho=datiInput(1);
                theta=datiInput(2);
                z=rho*exp(1i*theta);

    end % END SWITCH

    kInserito= datiInput(3);
    % CELL ARRAY CON TERMINI DELLE SERIE
    cellSeriePotenze={'z^k',... % GEOMETRICA
    '(z^k)/k ',... % ARMONICA
    '(z^k)/factorial(k)',... % ESPONENZIALE
    ' factorial(k)*(z^k)'}; % FATTORIALE (RECIPROCA ESP.)

    % MENU DI SCELTA DELLA SERIE
    index = menu('Scegli Serie S',cellSeriePotenze);

    % LA VARIABILE serie PUNTA ALLA STRINGA CHE INDICA LA SINGOLA
    % COMPONENTE DELLA SERIE SCELTA FRA QUELLE ELENCATE DAL CELL ARRAY
    % ATTRAVERSO LA FUNCTION menu() CHE ASSEGNA AD index L'INDICE DI
    % COMPONENTE RELATIVA ALLA SERIE
```

```
        serie=cellSeriePotenze{index};
        Sn=0; t=1; k=1;
        figure

        % CICLO WHILE
        % DISEGNA DI VOLTA IN VOLTA UN NUOVO VETTORE COLLEGATO AL PRECEDENTE
        % DOVE OGNI VETTORE t E' ORDINATAMENTE UNA COMPONENTE SUCCESSIVA
        % DELLA SERIE DI POTENZE ED E' VALUTATO SULLA RIDOTTA Sn DELL'ORDINE
        % PRECEDENTE CALCOLATA PER INCREMENTO DI t SU Sn
        while k<=kInserito
              Sn=Sn+t; % AGGIUNGE COMPONENTE A RIDOTTA DI ORDINE SUCCESSIVO
              % COMPONENTE DELLA SERIE CREATA COME UNA INLINE FUNCTION
              t=inline(serie,'z','k');
              t=t(z,k); % VALUTA LA COMPONENTE DELLA SERIE IN z E k
              quiver(real(Sn),imag(Sn),real(t),imag(t),0);

              % FUNCTION quiver()DISEGNA UN VETTORE A FRECCIA DI COMPONENTI
              % real(t) E imag(t), NEL PUNTO di coord real(Sn) e imag(Sn)
              axis equal;
              hold on
              k=k+1; % INCREMENTA k PER PASSARE A COMPONENTE SUCCESSIVA DELLA
SERIE
              pause(0.25); % CONSENTE L'ANIMAZIONE DEL DISEGNO
        end

        % CRITERI DI SELEZIONE USCITA PER GESTIRE CICLO WHILE
        % ATTRAVERSO UNA QUESTION DIALOG BOX
        uscita=questdlg('Vuoi uscire?','Continua...');

end % END WHILE

end % end function
```

Esempio d’uso




I...
real(z)=a
0.826
imag(z)=b
0.5
k (k>1)
8
OK
Cancel


Scegli Serie S
z^k
(z^k)/k
(z^k)/factorial(k)
factorial(k)*(z^k)

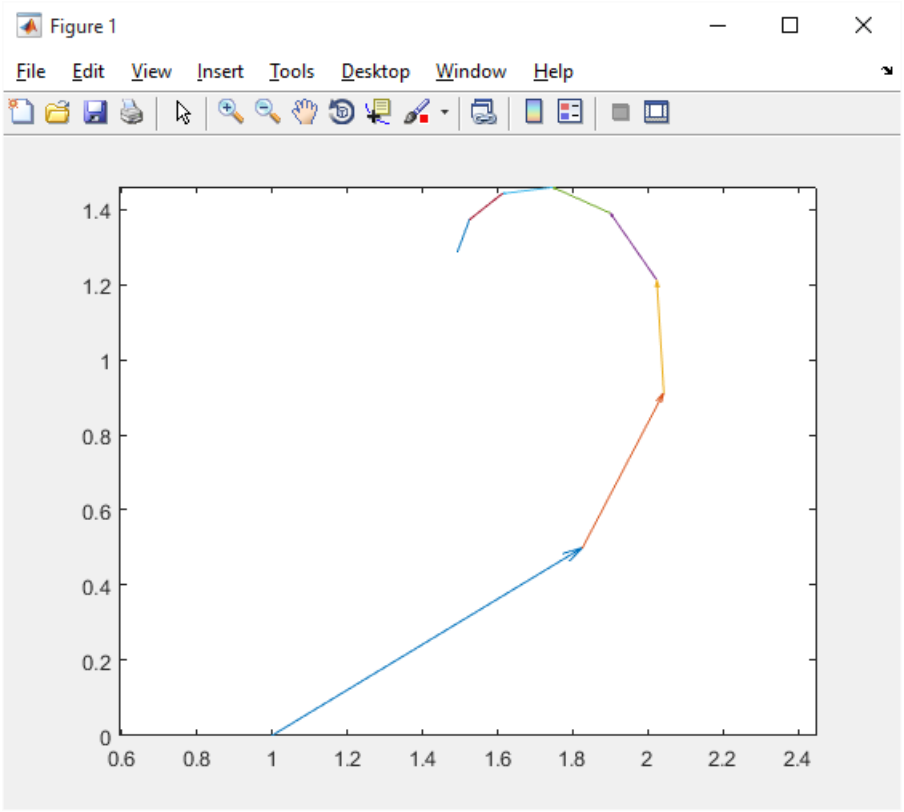


Figure 1
File Edit View Insert Tools Desktop Window Help
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0.6 0.8 1 1.2 1.4 1.6 1.8 2 2.2 2.4

Es. 29– Liv. 1 - Campo di convergenza

Visualizzare il campo di convergenza delle seguenti serie di potenze nel campo complesso: geometrica, armonica, esponenziale, ed ancora le serie

$$\sum_{n=0}^{\infty}\frac{z^n}{2^{2n}},\quad \sum_{n=1}^{\infty}\frac{(z+1)^n}{n2^n},\quad \sum_{n=1}^{\infty}\frac{(z-2)^n}{n^2 2^{2n}},\quad \sum_{n=0}^{\infty}(1+ni)z^n,\quad \sum_{n=1}^{\infty}(\log n)^2 z^n,\quad \sum_{n=0}^{\infty}\frac{5^n z^{3n}}{2n(2n+2)}.$$

Soluzione Proposta

Il Seguente programma studia la convergenza di una serie costruendo una sua Ridotta Sn di Ordine n inserito da Input Dialog Box come pure r Raggio di Convergenza del Cerchio.

Attraverso una Ridotta di Ordine n potremo infatti valutare la Convergenza della Serie, dal momento che quest'ultima tende a infinito, e sarebbe impossibile visualizzarla se non come un Piano Infinito.

Per la Selezione di una delle Serie Elencate, è stata sviluppata un'apposita Function `scelta_seriepotenze( )` proprio come per gli Elaborati delle Funioni Complesse

Tale function sarà soggetto di modifiche Simboliche o Numeriche in relazione alle necessità degli Elaborati futuri.

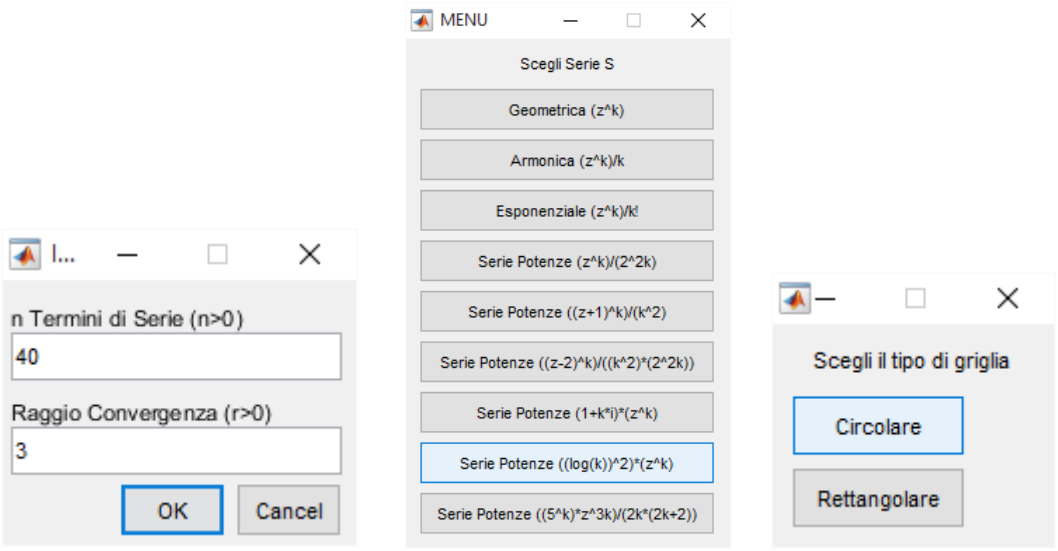
Dagli Output si puo vedere, che nelle zone del Cerchio di Convergenza in cui la Serie Assume Valori Bianchi, vuoti, questa non converge. Dalla seconda Curva che non produce Lacune si intuisce subito che essa è Convergente in tutto il Cerchio da noi definito

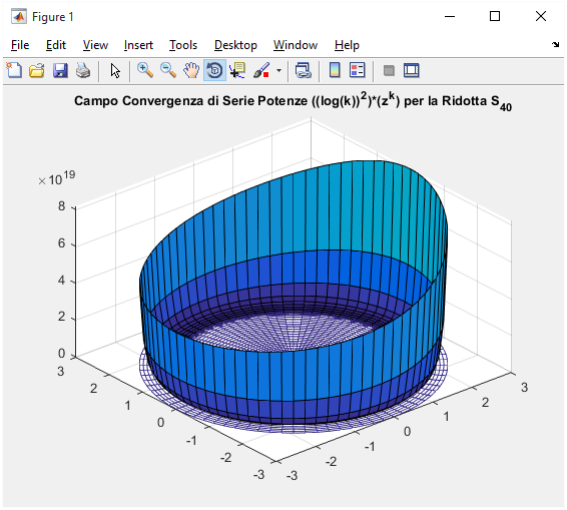
```
%   Matematica Applica e Computazionale
%   Convergenze di Serie Numeriche
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
%
% USO
%
% Lanciare da consolle il file
%
% >> convergenzeserie
%
% e vedere i calcoli che esegue

%%%%%%%
% Main
%%%%%%%

uscita='No';

f_convergenzaserie(uscita);
```

```
%   Matematica Applica e Computazionale
%
%   Convergenze di Serie Numeriche
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function f_convergenzaserie.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function f_convergenzaserie(uscita)

while strcmp(uscita,'No')
      % MENU' DI SCELTA DELL'ORDINE DELLA RIDOTTA Sn E DEL RAGGIO DI
CONVERGENZA
      campi={'n Termini di Serie (n>0)','Raggio Convergenza (r>0)'};
      titoloDLG='Input di Dati';
      numeroRighe=1;
      valoriDefault={'1','0.5'};
      datiInput = inputdlg(campi,titoloDLG,numeroRighe,valoriDefault);
      datiInput = char(datiInput);
      datiInput = str2num(datiInput);
      n = datiInput(1);
      r = datiInput(2);

      % RICHIAMA LA FUNCTION PER LA SELEZIONE DELLA SERIE SU CUI LAVORARE
      [ak, serieStringa] = scelta_seriepotenze(n);
```

```
      % MENU' DI SCELTA TIPO DI GRIGLIA
      sceltaGriglia=menu('Scegli il tipo di
griglia','Circolare','Rettangolare');

      switch sceltaGriglia
            case 1
                  % z PUNTA AD UNA GRIGLIA CIRCOLARE
                  z=(r*cplxgrid(n)); % griglia circolare di raggio r

            case 2
                  % z PUNTA AD UNA GRIGLIA RETTANGOLARE
                  [x,y]=meshgrid(linspace(-r,r,n));
                  z=complex(x,y);
      end % END SWITCH

      % restituisce i valori della ridotta di ordine n della serie prescelta
      % calcolata nei punti della griglia z
      % ak sono i coeff della ridotta
      Sn=polyval(ak,z);

      % DISEGNA IL MODULO DELLA RIDOTTA DELLA SERIE SCELTA abs(Sn)
      % IN FUNZIONE DI real(z) e imag(z)
      % il grafico del modulo di Sn è infatti sufficiente per avere un idea
      % della convergenza della serie
      figure
      surf(real(z),imag(z),abs(Sn));
      title(['Campo Convergenza di ' serieStringa...
      ' per la Ridotta S_{' num2str(n) '}'],'FontSize',10)
      hold on; axis tight; V=axis;

      % DISEGNA LA GRIGLIA SCELTA AL DI SOTTO DEL GRAFICO DELLA SERIE
      mesh(real(z),imag(z),V(5)*ones(size(z)))

      % CRITERI DI SELEZIONE USCITA PER GESTIRE CICLO WHILE
      % ATTRAVERSO UNA QUESTION DIALOG BOX
      uscita=questdlg('Vuoi uscire?','Continua...');

end % END WHILE

end % END FUNCTION
```

Esempio d'uso





Es. 30– Liv. 2 – Sym studio convergenza

Studiare le serie elencate nell'esercizio precedente mediante il *Symbolic Math Toolbox* individuandone il campo di convergenza e l'eventuale somma. Giustificare i risultati ottenuti

Soluzione proposta

Il Programma è un'elaborazione ibrido in Numerico e Simbolico che studia la Convergenza di una Serie di Potenze, scelta tramite un'apposita Function `symscelta_serie()`, che restituisce il Valore di Partenza k_inizio della Serie, una Stringa col nome di questa, ed inoltre restituisce sia a, la Function Inline, rappresentativa della k-sima Componente della Serie, sia lo stesso termine ak costruito in Symbolic: a serve a calcolare il Raggio di Convergenza del Cerchio, possibile solo in Ambiente Numerico, mentre ak consente di calcolare la Sommatoria Simbolica tendente a Infinito tramite la Matlab Function `symsum()`.

Oltre la Curva visualizzata ad AXIS, l'Output mostra una Finestra `text()` dei valori di Serie e Raggio Convergenza calcolati, e nel caso di Divergenza, mostra una Error Dialog Box che annuncia il caso di Limite Infinito

NOTA: Poichè `symsum()`, per Serie non semplici, può restituire valori di Convergenza distribuiti su intervalli piecewice, generando un Errore, si è deciso di implementare un DOPPIO TRY CATCH ad Innesto.

Il primo TRY prova a visualizzare la Curva Complessa, ma se si genere un risultato piecewice allora si salta al CATCH che implementa proprio il secondo TRY, questo tenta attraverso la Matlab Function `find()`, di estrarre i risultati contenuti tra i simboli " ' " e ']', ma se il risultato è molto complesso, il secondo CATCH fa si che nella Finestra `text()` oltre i risultati Matematici, sia specificato anche il motivo per cui il Disegno non puo' essere visualizzato.

```
%   Matematica Applica e Computazionale
%
%   Convergenze di Serie Numeriche
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
%
% USO
%
% Lanciare da consolle il file
%
% >> symstudioconvergenza
%
% e vedere i calcoli che esegue

%%%%%%%
% Main
%%%%%%%

uscita='No';
syms z real
syms k positive
fsym_studioconvergenza(z,k,uscita);
```

```
%   Matematica Applica e Computazionale
%
%   Convergenze di Serie Numeriche
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function fsym_studioconvergenza.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function fsym_studioconvergenza(z,k,uscita)

while strcmp(uscita,'No')
      % MENU SCELTA SERIE
      % RESTITUISCE IL TERMINE DELLA SERIE a DEFINITO COME INLINE FUNCTION
      % ED IL TERMINE DELLA SERIE ak DEFINITO COME SYMBOLIC STRING
      [a, ak, k_inizio, serieString] = sym_sceltaserie(z,k);

      % LIMITE DEL RAPPORTO
      % CALCOLO DEL RAGGIO DI CONVERGENZA COME LIMITE DEL RAPPORTO FRA UN
      % TERMINE DELLA SERIE E IL SUO SUCCESSIVO
      r_conv=limit(a(k)/a(k+1),k,inf); %RAGGIO CONVERGENZA
      S=simplify(symsum(ak,k,k_inizio,inf)); %SOMMATORIA DELLA SERIE
      cellRisultato={['Raggio di Convergenza:' char(r_conv)] ,...
      ['Sommatoria Serie:' char(S)]};

      % se il raggio di convergenza è infinito ...
      if strcmp(char(r_conv),'Inf')
            errordlg('LA FUNZIONE E'' DIVERGENTE!
(r_conv=Inf)','Divergenza Serie');
      else
```

```
            % COSTRUTTO TRY-CATCH
            % UTILIZZA UN TRY-CATCH PERCHE' SPESSO I RISULTATI DELLA
            % SOMMATORIA HANNO UN VALORE FRAMMENTATO IN PIU' INTERVALLI, E
            % CIO' IMPEDISCE IL DISEGNO DELLA CURVA CHE MOSTRA LA
            % CONVERGENZA DELLA SERIE DI POTENZE NEL CASO IN CUI CIO' SI
            % VERIFICHI, SI ENTRA NEL CATCH DOVE IL RELATIVO BLOCCO DI
            % ISTRUZIONI UTILIZZA FIND PER TROVARE I "TOKENS" CHE
            % DELIMITANO IL RISULTATO
            try % PROVA A DISEGNARE LA SOMMATORIA
                  ezsurf(abs(S),[-r_conv r_conv -r_conv r_conv])
                  title(serieString)
            catch % SE NON CI RIESCE...

                  try % PROVA ALLORA A....
                        S=char(S); % CONVERTIRE SOMMATORIA SERIE DA SYMOLIC
A STRING
                        j1=find(S==',',1,'last'); % TROVARE IL TOKEN A SX
DEL RISULTATO
                        j2=find(S==']',1,'last'); % TROVARE IL TOKEN A DX
DEL RISULTATO
                        S=S((j1+1):(j2-1)); % ESTRAE RISULTATO DELIMITATO
TAI TOKENS
                        S=sym(S); % RICONVERTIRE IL RISULTATO IN SIMBOLICO
                        r_conv=str2num(char(r_conv));
                        % DISEGNA
                        ezsurf(abs(S),[-r_conv r_conv -r_conv r_conv])
                        title(serieString)

                        catch % SE NON RIESCE NEANCHE STAVOLTA...
                        cellRisultato(3)={'ATTENZIONE! Troppe limitazioni
in Matlab'...
                        ' per Disegnare Sommatoria'};

                        % AGGIUNGE DINAMICAMENTE UNA TERZA CELLA CONTENENTE
                        % STRINGA DI MESSAGGIO DIAGNOSTICO AL CELL ARRAY
                        % cellRisultato

                  end % END TRAY-CATCH
            end % END TRAY-CATCH
      end % END IF-ELSE
```

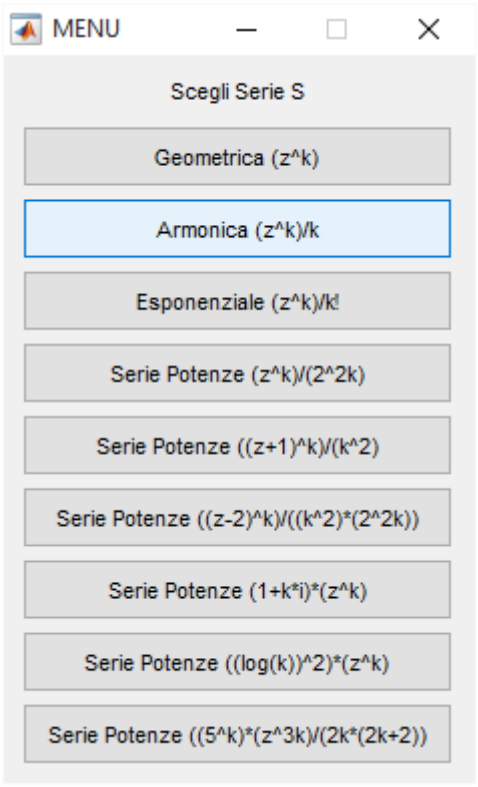
```
        % STAMPA A SCHERMO I RISULTATI PER IL RAGGIO DI CONVERGENZA
        % IL VALORE DELLA SOMMATORIA
        % E PER EVENTUALE ERRORE DI VISUALIZZAZIONE DEL GRAFICO
        figure
        text(.5,.5,cellRisultato,'FontSize',12,...
        'HorizontalAlignment','center','Color','b','FontWeight','bold');
        axis off

        % QUESTION DIALOG BOX PER LA SCELTA DI USCITA
        uscita=questdlg('Vuoi uscire?','Continua...');

end % END WHILE

end % END FUNCTION
```

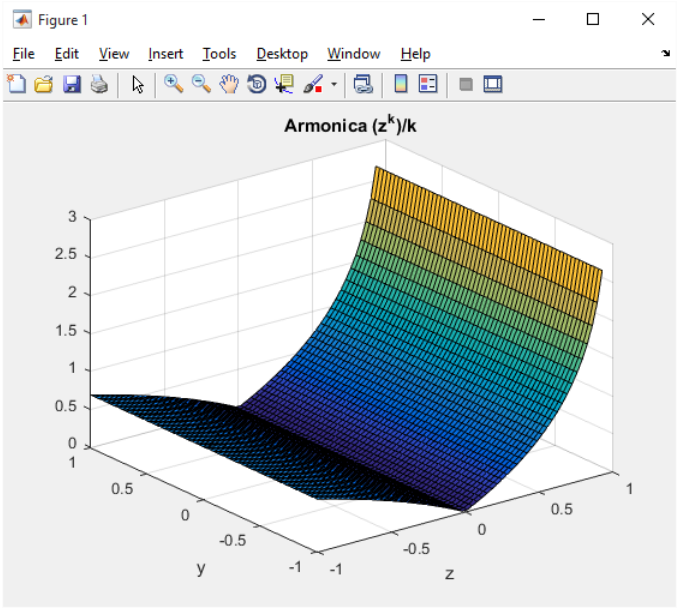
```
%   Matematica Applica e Computazionale
%
%   Convergenze di Serie Numeriche
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function sym_sceltaserie.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% GENERA UN MENU' DI SERIE DA SCEGLIERE E RESTITUISCE SIA
% UNA FUNZIONE RAPPRESENTATE IL TERMINE GENERICO DELLA SERIE SIMBOLICA
% SIA UNA STRINGA CHE RAPPRESENTA IL NOME DELLA SERIE

function [a,ak,k_inizio,serieString] = sym_sceltaserie(z,k)
% CELL ARRAY DI STRINGHE PER TITOLI BOTTONI DEL MENU' FUNZIONE
cellSerie={'Geometrica (z^k)';...
'Armonica (z^k)/k';...
'Esponenziale (z^k)/k!';...
'Serie Potenze (z^k)/(2^2k)';...
'Serie Potenze ((z+1)^k)/(k^2)';...
'Serie Potenze ((z-2)^k)/((k^2)*(2^2k))';...
'Serie Potenze (1+k*i)*(z^k)';...
'Serie Potenze ((log(k))^2)*(z^k)';...
'Serie Potenze ((5^k)*(z^3k)/(2k*(2k+2))'};

% MENU DI SCELTA DELLA SERIE
serieScelta = menu('Scegli Serie S',cellSerie);
```

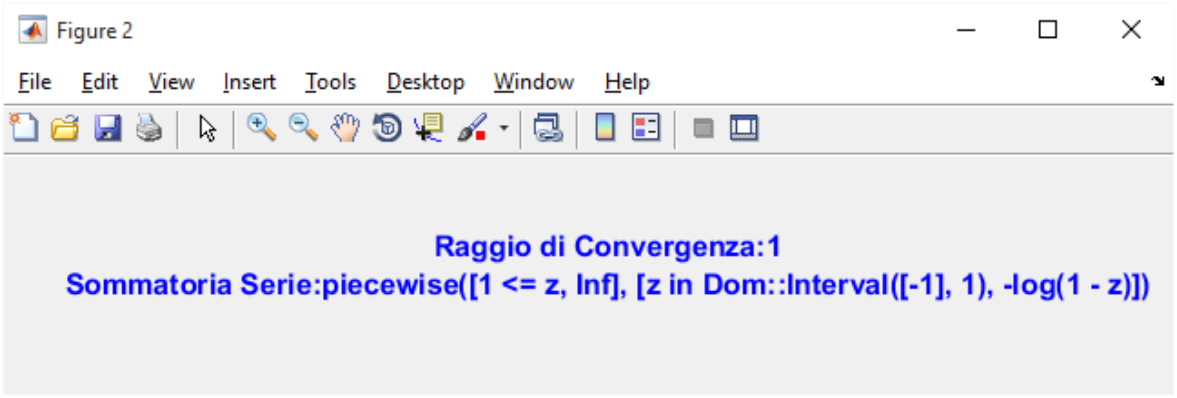
```
% A SECONDA DELLA SERIE SCELTA ASSEGNO AD ak IL VALORE DEL
% IL VALORE DEL TERMINE GENERICO DELLA SERIE DI POTENZE
% DEFINITO COME UNA FUNZIONE IN (z,k)
switch serieScelta
      case 1
            % TERMINE ak DELLA SERIE z^k
            a=inline('k/k');
            ak=a(k)*(z^k);
            k_inizio=0;

      case 2
            % TERMINE ak DELLA SERIE (z^k)/k
            a=inline('1/k');
            ak=a(k)*(z^k);
            k_inizio=1;

      case 3
            % TERMINE ak DELLA SERIE (z^k)/k!
            a=inline('1/gamma(k+1)');
            ak=a(k)*(z^k);
            k_inizio=0;

      case 4
            a=inline('1/(4^k)');
            ak=a(k)*(z^k);
            k_inizio=0;

      case 5
            a=inline('1/(k^2)');
            ak=a(k)*((z+1)^k);
            k_inizio=1;

      case 6
            % ((z-2)^k)/((k^2)*(2^(2*k)))
            a=inline('1/((k^2)*4^k)');
            ak=a(k)*((z-2)^k);
            k_inizio=1;

      case 7
            % (1+k*1i)*(z^k)%
            a=inline('1+k*1i');
            ak=a(k)*(z^k);
```

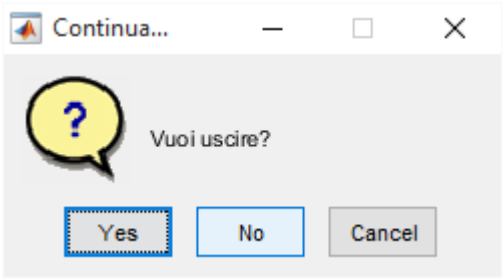
```
            k_inizio=0;

      case 8
            % ((log(k)^2)*(z^k))
            a=inline('log(k)^2');
            ak=a(k)*(z^k);
            k_inizio=1;

      case 9
            % ((5^k)*(z^(3*k)))/((2*k)*(2*k+2))
            a=inline('1/(2*k*(2*k+2))');
            ak=a(k)*(z^k);
            k_inizio=0;
end

% ASSEGNA A serieString IL NOME DELLA SERIE SCELTA INDICIZZANDO
% CORRETTAMENTE IL CELL ARRAY DEFINITO cellSerie
serieString=cellSerie{serieScelta};
end
```

Esempio d'uso









Comprortamento sulla frontiera del campo di convergenza

Es. 31– Liv. 1 - Campo di convergenza

Visualizzare il comportamento sulla frontiera delle seguenti serie di potenze nel campo complesso: geometrica, armonica, esponenziale, ed ancora le serie

$$\sum_{n=0}^{\infty}\frac{z^n}{2^{2n}},\quad \sum_{n=1}^{\infty}\frac{(z+1)^n}{n2^n},\quad \sum_{n=1}^{\infty}\frac{(z-2)^n}{n^2 2^{2n}},\quad \sum_{n=0}^{\infty}(1+ni)z^n,\quad \sum_{n=1}^{\infty}(\log n)^2 z^n,\quad \sum_{n=0}^{\infty}\frac{5^n z^{3n}}{2n(2n+2)}.$$

Soluzione proposta

L'Elaborato in Symbolic, in seguito alla Selezione della Serie da Menu Grafico, e all'impostazione dei parametric di Input quale Ordine di Ridotta n e valore del Raggio di Convergenza r tramite Input Dialog Box, inizialmente calcola la Ridotta per Visualizzarla Completamente, successivamente visualizza la parte di Curva Complessa relativa solo ai Complessi z Più Esterni sulla Griglia Circolare per far sì che sia visualizzabile unicamente il Comportamento sulla Frontiera.

Il programma calcola la Convergenza in tutto il cerchio sia per Sommatoria dei Termini Derivati sia per Derivazione della Serie Completa, visualizza entrambi i risultati in text(), ma per visualizzare la Convergenza su tutto il Cerchio, si stampa a `ezsurf()` solo il risultato ottenuto col primo Metodo.

Per estrarre il risultato da un eventuale piecewice, è stata implementata una Function apposta chiamata `f_estrai_seriesym()`, ai commenti il compito di spiegarla in dettaglio.

Per la scelta della serie, è stata invece implementata una modifica della Function di Selezione, `symscelta_seriepotenze2()`

```
%   Matematica Applica e Computazionale
%
%   Convergenza di frontiera
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
%
% USO
%
% Lanciare da consolle il file
%
% >> convergenzafrontiera
%
% e vedere i calcoli che esegue

%%%%%%%
% Main
%%%%%%%

clc % pulisce schermo matlab

uscita='No';
f_convergenzaefrontiera(uscita)
```

```
%   Matematica Applica e Computazionale
%
%   Convergenza di frontiera
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function f_convergenzaefrontiera.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function f_convergenzaefrontiera(uscita)

while strcmp(uscita,'No')
      syms somSerie real
      syms k positive

      % MENU' SCELTA DELL'ORDINE DELLA RIDOTTA Sn E DEL RAGGIO DI
CONVERGENZA
      campi={'n Termini di Serie (n>0)','Semiampiezza (r>0)'};
      titoloDLG='Input di Dati';
      numeroRighe=1;
      valoriDefault={'1','0.5','5'};
      datiInput = inputdlg(campi,titoloDLG,numeroRighe,valoriDefault);
      datiInput = char(datiInput);
      datiInput = str2num(datiInput);
      n = datiInput(1);
      r = datiInput(2);

      % RICHIAMA LA FUNCTION PER LA SELEZIONE DELLA FUNZIONE DA DISEGNARE
      [ak, a_t, i_inizio, serieStringa] = scelta_seriepotenze2(n);
```

```
      % SERIE - SOMMATORIA COMPLETA
      somSerie=simplify(symsum(a_t,k,i_inizio,inf));

      % DERIVATA DEL TERMINE GENERICO DELLA SERIE
      atDerivata=simplify(diff(a_t));

      % TRY CATCH - A CAUSA DELLE DIFFERENZE MAPLE/MUPAD
      try
            % SOMMATORIA SUL TERMINE DERIVATO
            som_atDerivata=symsum(atDerivata,k,i_inizio,inf);
            derivSomSerie=diff(somSerie); % DERIVATA DELLA SOMMATORIA
DELLA SERIA
            catch
            errordlg('ERRORE MATLAB!',...
            'ERRORE PER LE DIFFERENZE DI COMPILAZIONE FRA MUPAD E MAPLE');
      end % END TRY

      %%%%%%%%%%% FUNCTION f_estray_seriesym() %%%%%%%%%%%
      % SERVE AD ESTRARRE IL RISULTATO DI UNA SERIE SIMBOLICA NEL MOMENTO
      % IN CUI ESSO NON E' UN VALORE UNICO, MA E' VALUTATO SU PIU'
      % INTERVALLI RIASSUNTI SU DI UNA SOLA STRINGA. TALE FUNCTION
      % ESTRAE DA TALE STRINGA L'UNICO RISULTATO CHE A NOI INTERESSA PER
      % IL DISEGNO. NEL CASO IN CUI NELLE VARIABILI SIANO CONTENUTI GIA' I
      % VALORI DESIDERATI, TALE FUNCTION NON ALTERA ULTERIORMENTE IL LORO
      % VALORE, MA RESTITUISCE SEMPRE QUELLO CORRETTO.
      % COSTRUTTO TRY
      % LO USIAMO PER APPLICARE L'EFFETTO DI f_estrai_seriesym()
      % SOLO SE NECESSARIO AL NOSTRO CASO
      try
            som_atDerivata=f_estrai_seriesym(som_atDerivata);
            derivSomSerie=f_estrai_seriesym(derivSomSerie);
      end

      % DEFINISCE I NUMERI COMPLESSI IN z ATTRAVERSO GRIGLIA COMPLESSA
CIRCOLARE
      z=(r*cplxgrid(n));

      % DISEGNA 2 GRAFICI, UNO PER TUTTI I PUNTI DELLA RIDOTTA,
      % L'ALTRO PER GLI UNICI PUNTI CORRISPONDENTI
      % A QUELLI DELLA FRONTIERA DELLA SERIE
      % DELLA PARTE REALE DEI VETTORI COMPLESSI real(z_fr) E DELLA LORO
PARTE
```

```
% IMMAGINARIA imag(z_fr)
%
% RICORDA: LA SERIE CONVERGE NEL CERCHIO E SULLA FRONTIERA, MA AL
% PIU', NON CONVERGE DOVE PRESENTA PICCO SULLA FRONTIERA
% RIDOTTA COMPLETA
% VALUTA IL POLINOMIO DELLA RIDOTTA UNICAMENTE SU TUTTI I PUNTI DELLA
% SERIE
RidottaCompleta=polyval(ak,z);
figure
surf(real(z),imag(z),abs(RidottaCompleta));
title(['Campo Convergenza di ' serieStringa...
' per la Ridotta S_{' num2str(n) '}'],'FontSize',10)
hold on; axis tight; V=axis;
% DISEGNA LA GRIGLIA SCELTA AL DI SOTTO DEL GRAFICO DELLA SERIE
mesh(real(z),imag(z),V(5)*ones(size(z)))

%%% FRONTIERA %%%
% ESTRAE I VETTORI COMPLESSI DI FRONTIERA E QUELLI DELLA RIGA
PRECEDENTE
z_fr=z(end-1:end,:);

% VALUTA IL POLINOMIO DELLA RIDOTTA UNICAMENTE SUI PUNTI DI FRONTIERA
Frontiera=polyval(ak,z_fr);
figure
surf(real(z_fr),imag(z_fr),abs(Frontiera));
title(['Campo Convergenza di ' serieStringa...
' per la Ridotta S_{' num2str(n) '}'],'FontSize',10)
hold on; axis tight; V=axis;

% DISEGNA LA GRIGLIA SCELTA AL DI SOTTO DEL GRAFICO DELLA SERIE
mesh(real(z),imag(z),V(5)*ones(size(z)))

% SOMMATORIA TERMINE DERIVATO
figure
ezsurf(abs(som_atDerivata),[-r r -r r])
titolo=['Sommatoria per Derivazione Termine a Termine di : '
serieStringa];
title(titolo);
```

```
        % STAMPA AD AXIS I RISULTATI DELLA SOMMATORIA DEL TERMINE DERIVATO E
        % DELLA SOMMATORIA DELLA SERIE DERIVATA
        figure
        cellRisultato={['Sommatoria del Termine Derivato'...
        ' somm(D[ak]) = ' char(som_atDerivata)],...
        ['Sommatoria della Serie Derivata'...
        ' somm(D[somm(ak)]= ' char(derivSomSerie)]};
        text(.5,.5,cellRisultato,'FontSize',10,...
        'HorizontalAlignment','center','Color','b','FontWeight','bold');
        axis off

        % CRITERI DI SELEZIONE USCITA PER GESTIRE CICLO WHILE
        % ATTRAVERSO UNA QUESTION DIALOG BOX
        uscita=questdlg('Vuoi uscire?','Continua...');

end % END WHILE

end % END FUNCTION
```

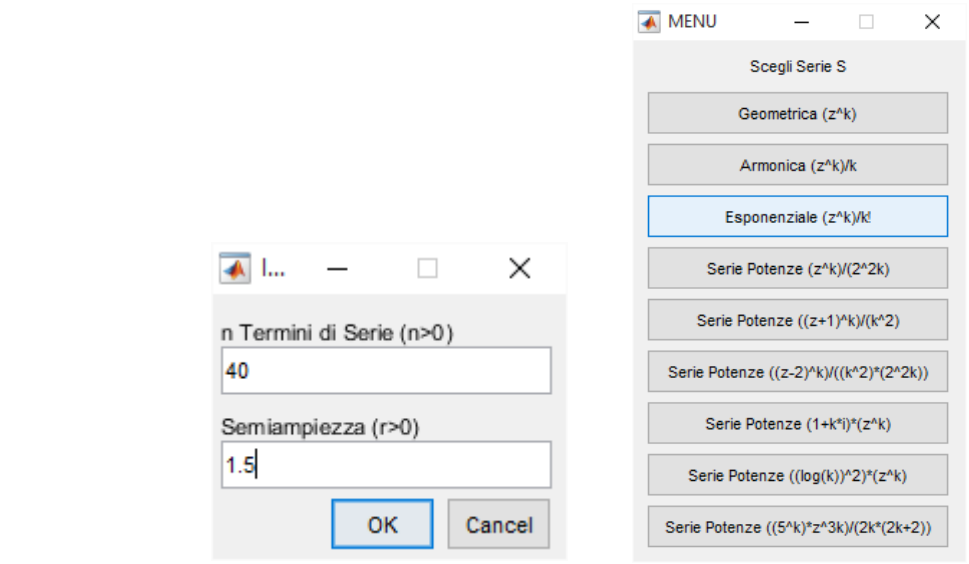
```
%   Matematica Applica e Computazionale
%
%   Convergenza di frontiera
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function f_estrai_seriesym.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% LA FUNCTION EFFETTUA L'ESTRAZIONE DEL RISULTATO SIMBOLICO DI UNA SERIE
% QUALORA TALE RISULTATO FOSSE UNA STRINGA CONTENENTE INTERVALLI piecewise
% E NON UN VALORE SIMBOLICO UNICO. IN CASO CONTRARIO, NON HA ALCUN EFFETTO
% SUL RESTO DEL PROGRAMMA, POICHE' QUESTO INSERISCE LA FUNCTION IN UN
% TRY CATCH
function risultato= f_estrai_seriesym(S)

X=char(S); % CONVERTIRE LA SOMMATORIA DELLA SERIE DA SYMOLIC A STRING
j1=find(X==',',1,'last'); % TROVARE IL TOKEN A SX DEL RISULTATO
j2=find(X==']',1,'last'); % TROVARE IL TOKEN A DX DEL RISULTATO
X=X((j1+1):(j2-1)); % ESTRAE RISULTATO COMPRESO FRA I TOKEN
risultato=sym(X); % RICONVERTIRE IL RISULTATO IN SIMBOLICO

end % END FUNCTION
```

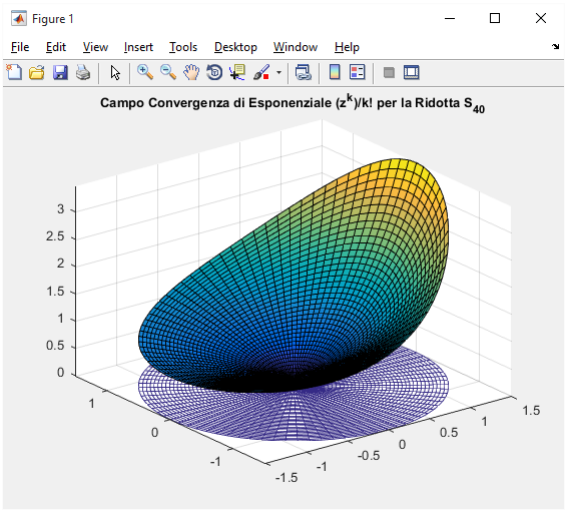
```
%   Matematica Applica e Computazionale
%
%   Convergenza di frontiera
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function scelta_seriepotenze2.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% GENERA UN MENU' DI SERIE DA SCEGLIERE E RESTITUISCE SIA
% LA SERIE DEFINITIA ATTRAVERSO UNA ANONIMOUS FUNCTION
% SIA UNA STRINGA CHE RAPPRESENTA IL NOME DELLA SERIE
function [ak,a_termine,i_inizio,serieString] = scelta_seriepotenze2(n)

% ak: coeff della ridotta
syms ak z real
syms k positive

% CELL ARRAY DI STRINGHE PER TITOLI BOTTONI DEL MENU' FUNZIONE
cellSerie={'Geometrica (z^k)';...
'Armonica (z^k)/k';...
'Esponenziale (z^k)/k!';...
'Serie Potenze (z^k)/(2^2k)';...
'Serie Potenze ((z+1)^k)/(k^2)';...
'Serie Potenze ((z-2)^k)/((k^2)*(2^2k))';...
'Serie Potenze (1+k*i)*(z^k)';...
'Serie Potenze ((log(k))^2)*(z^k)';...
'Serie Potenze ((5^k)*z^3k)/(2k*(2k+2))'};
```

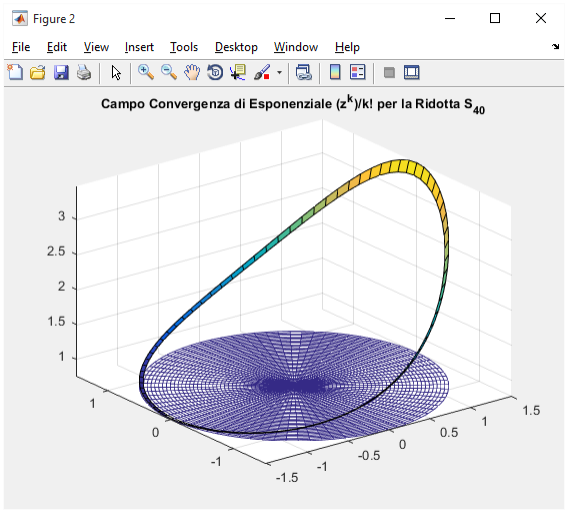
```
% MENU DI SCELTA DELLA SERIE
serieScelta = menu('Scegli Serie S',cellSerie);
% A SECONDA DELLA SERIE SCELTA ASSEGNO AD ak IL VALORE DEL
% GENERICO TERMINE DELLA SERIE

switch serieScelta

      case 1
            % TERMINE ak DELLA SERIE GEOMETRICA
            ak=ones(1,n);
            ak=[0 ak];
            a_termine=z^k;
            i_inizio=0;

      case 2
            % TERMINE ak DELLA SERIE ARMONICA
            ak=(1./(1:n));
            a_termine=(z^k)/k;
            i_inizio=1;

      case 3
            % TERMINE ak DELLA SERIE ESPONENZIALE
            ak=cumprod(1./(1:n));
            ak=[0 ak];
            % UTILIZZO prod() ANZICHE' gamma(n+1) POICHE' ezsurf() NON SI
            % PRESTA A DISEGNARE TALE FUNCTION
            a_termine=(z^k)/prod(1:n);
            i_inizio=0;

      case 4
            % TERMINE ak DELLA SERIE DI POTENZE (z^k)/(2^2k)
            ak=1./(2.^(2.*(1:n)));
            ak=[0 ak];
            a_termine=(z^k)/(2^(2*k));
            i_inizio=0;

      case 5
            % TERMINE ak DELLA SERIE DI POTENZE ((z+1)^k)/(k^2)
            ak=1./((1:n).*2.^(1:n));
            a_termine=((z+1)^k)/(k^2);
            i_inizio=1;
```

```
        case 6
            % TERMINE ak DELLA SERIE DI POTENZE ((z-2)^k)/((k^2)*(2^2k))
            ak=1./(((1:n).^2).*4.^(1:n));
            a_termine=((z-2)^k)/((k^2)*(2^(2*k)));
            i_inizio=1;

        case 7
            % TERMINE ak DELLA SERIE DI POTENZE (1+k*i)*(z^k)
            ak=(1+(1:n).*1i);
            ak=[0 ak];
            a_termine=(1+k*1i)*(z^k);
            i_inizio=0;

        case 8
            % TERMINE ak DELLA SERIE DI POTENZE ((log(k))^2)*(z^k)
            ak=((log(1:n)).^2);
            a_termine=((log(k))^2)*(z^k);
            i_inizio=1;

        case 9
            % TERMINE ak DELLA SERIE DI POTENZE ((5^k)*z^3k)/(2k*(2k+2))
            ak=(5.^(1:n))./(2.*(1:n).*(2.*(1:n)+2));
            ak=[0 ak];
            a_termine=((5^k)*(z^(3*k)))/((2*k)*((2*k)+2));
            i_inizio=0;

end % END SWITCH

% INVERTE L'ORDINE DI DISPOSIZIONE DELLE COMPONENTI DI ak PER RENDERLE
% UTILIZABILI DA polyval()
ak=fliplr(ak);

% ASSEGNA A serieString IL NOME DELLA SERIE SCELTA INDICIZZANDO
% CORRETTAMENTE IL CELL ARRAY DEFINITO cellSerie
serieString=cellSerie{serieScelta};

end % END FUNCTION
```

Esempio d'uso







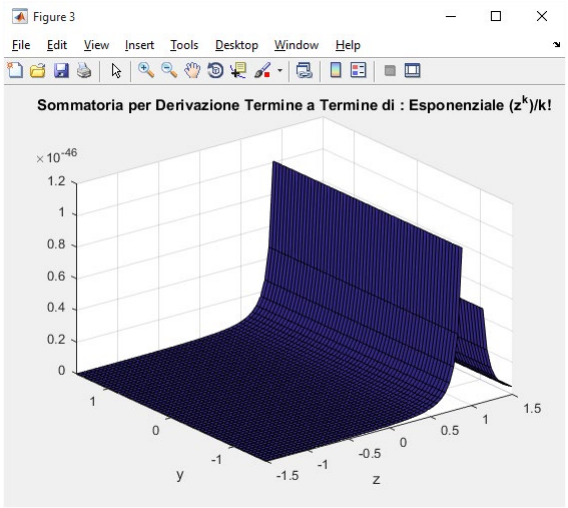



Figure 4
File Edit View Insert Tools Desktop Window Help
Sommatoria del Termine Derivato somm(D[ak]) = 1/(81591528324789768379554852130119379035998493081 6*(z - 1)^2)
Sommatoria della Serie Derivata somm(D[somm(ak)]= 1/(81591528324789768379554852130119379035998493081 6*(z - 1)^2)

Interpolazione trigonomoetrica

Es. 33– Quiz – Interpolazione 1

Come mai, nonostante il Teor. di esistenza ed unicità del polinomio trigonometrico interpolante, nell'esempio (N=4, 5 nodi di interpolazione - nel file Interpolazione Trigonometrica (parte I)) si trovano due polinomi interpolanti Q e T diversi?

Soluzione proposta

Il Teorema di Esistenza ed Unicità di Lagrange, per un Polinomio Trigonometrico Interpolante Q(z), afferna che, assegnati N+1 Nodi Complessi zk distinti in [a,b[, cioè tali che real(zk) ≠ real(zj) (Ascisse dei Nodi Distinti), esiste un unico Polinomio Trigonometrico Interpolante Q(z) di grado
al piu' N, tale che Q(zk)=yk.
Abbiamo però riscontrato che è possibile definire 2 Polinomi Q(z) e T(z)
Ciò accade perché, pur trattandosi di 2 Polinomi Differenti, essi vengono ricavati l'uno dall'altro con variazioni di calcolo. Dagli Output però si evince, che le Curve realizzate sono REALMENTE 2 CURVE DIVERSE. Come mai? Ebbene, i 2 polinomi hanno diversi dati in input!!
Algoritmo Q: L'Algoritmo che mira alla costruzione di Q, Polinomio Scalare dalla Forma Algebrica, prende in input le Ascisse xk e le Ordinate yk dei Nodi di Interpolazione z, e per la Costruzione e Risoluzione del Sistema Lineare lavora solo su queste coordinate, creando una curva interpolante ma NON PRESICISA.
Algoritmo T: Non effettua un'interpolazione sulle Coordinate dei Nodi, ma Proprio sui Nodi Complessi z, che è l'Input su cui si costruiscono le Matrici A e c del Sistema Lineare assieme al Fattore Moltiplicativo –N/2 che Transla L'Intervallo Opportunamente. Tali z Complessi sono Vettori, e da ciò si battezza Forma Vettoriale appunto quella del Polinomio T. Si nota come la curva di Tale Polinomio Vettoriale INTERPOLA PERFETTAMENTE la

Funzione nei Nodi Complessi, Diversamente dal Q(z). Il Polinomio T(z) altro non è che una Seconda Particolare

Forma in cui si può esprimere il Polinomio Q(z), utilizzata solo nel caso in cui il Grado dei Polinomi è pari, e quindi quando il si ha un Numero di Nodi Dispari, poiché solo in Tal caso è possibile la Translazione in N/2 di Semiampiezza.

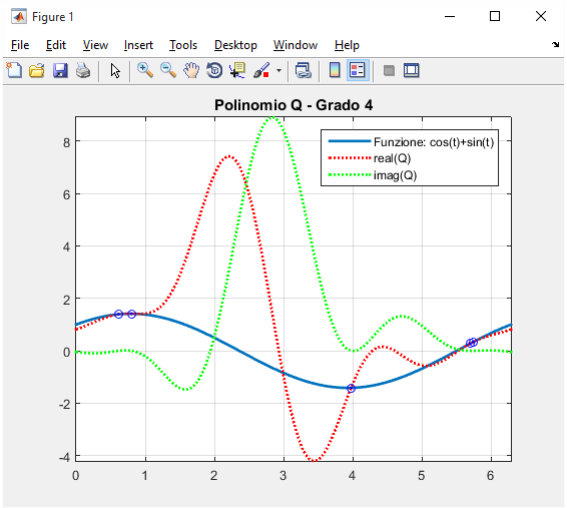
Inoltre, come si può notare dagli Output, il Polinomio Q restituisce Valori Complessi, poiché la sua Parte Immaginaria è una Curva 2D Irregolare e Diversa da 0, mentre il Polinomio T restituisce evidentemente Valori Reali, dal momento che ha Parte Immaginaria Nulla.

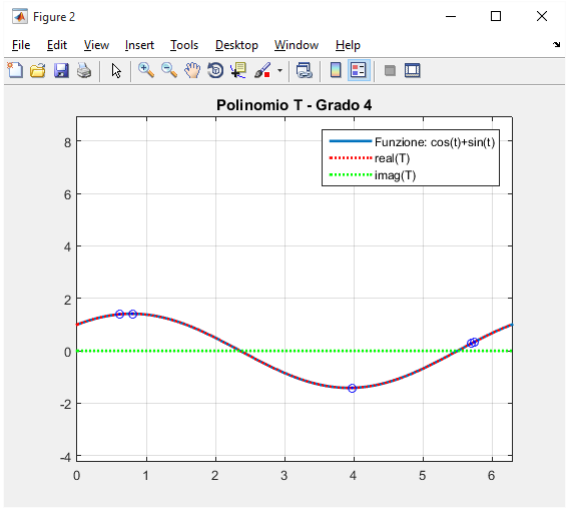
```
%   Matematica Applicata e Computazionale
%
%   QUIZ Interpolazione Trigonometrica
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
%
% USO
%
% Lanciare da consolle il file
%
% >> quizinterpolazione1
%
% e vedere i calcoli che esegue

%%%%%%%
% Main
%%%%%%%

% QUIZ
% Come mai, nonostante il Teor. di esistenza ed unicità del polinomio
% trigonometrico interpolante, nell'esempio (N=4, 5 nodi di interpolazione
% - nel file Interpolazione Trigonometrica (parte I)) si trovano due
% polinomi interpolanti Q e T diversi?

clear all
funzione=inline('cos(t)+sin(t)');
Npunti=4+1.*round(rand(1)); % GENERA ALLE VOLTE 4 E A VOLTE 5
N=Npunti-1; % DEFINISCE IL GRADO DEL POLINOMIO
k=0:N;
```

```
% DATI PER IL POLINOMIO INTERPOLANTE Q(z)
xk=2*pi*rand(Npunti,1); % GENERA NODI DI ASCISSA
yk=funzione(xk); % CALCOLA RELATIVE ORDINATE
A=exp(1i*xk*k); % MATRICE DEL SISTEMA LINEARE
c=A\yk; % MATRICE DEI COEFFICIENTI DEL POLINOMIO

% DATI PER LA FUNZIONE REALE EFFETTIVA f(x)
x=linspace(0,2*pi,199); % ASCISSE DELLA FUNZIONE
y=funzione(x); % ORDINATE DELLA FUNZIONE
Q=polyval(flipud(c),exp(1i*x)); % POLINOM. TRIGONOMETRICO INTERPOLANTE Q

% DISEGNA LA FUNZIONE y=cos(x)+sin(x), LA PARTE REALE DEL POLINOMIO Q
% real(Q) E QUELLA IMMAGINARIA imag(Q)
figure
plot(x,y,x,real(Q),'r:', x, imag(Q),'g:','LineWidth', 2);
hold on
grid
axis tight
h=axis; % PER SETTARE GLI ASSI DEL 2° FIGURE IN EGUAL MODO

% DISEGNA I NODI DI INTERPOLAZIONE CON UN CERCHIO E UN PUNTO
plot(xk,yk,'.b',xk,yk,'ob')

% LEGENDA
legend(['Funzione: ' char(funzione)] , 'real(Q)' , 'imag(Q)');
title(['Polinomio Q - Grado ' num2str(N)]);

if(mod(N,2)==0)
      zk=exp(1i*xk);
      A=exp(1i*xk*k);
      c=A\(yk.*zk.^(N/2));

      % DATI PER LA FUNZIONE REALE EFFETTIVA f(x)
      x=linspace(0,2*pi,199);
      y=funzione(x);
      z=exp(1i*x);
      T=z.^(-N/2).*polyval(flipud(c),exp(1i*x));

      % DISEGNA LA FUNZIONE y=cos(x)+sin(x) LA PARTE REALE DEL POLINOMIO T
      % real(T) E QUELLA IMMAGINARIA imag(T)
      figure
      plot(x,y,x,real(T),'r:', x, imag(T),'g:', 'LineWidth', 2);
```

```
        hold on
        axis tight
        ylim([h(3) h(4)]) %SETTIAMO ASSI SIMILI
        grid

        % DISEGNA I NODI DI INTERPOLAZIONE CON UN CERCHIO E UN PUNTO
        plot(xk,yk,'.b',xk,yk,'ob');

        % LEGENDA
        legend(['Funzione: ' char(funzione)] , 'real(T)' , 'imag(T)');
        title(['Polinomio T - Grado ' num2str(N)]);

end % END IF
```

Esempio d'uso





Es. 37– Liv. 1 - Interpolazione Line Space

Per le funzioni (§§) costruire efficientemente, al variare del numero N dei nodi, i polinomi trigonometrici interpolanti Q e T generando i nodi di interpolazione come:

1) **xi=lnspace(…).**
2) **xi=rand(…).**

Confrontare graficamente tra lori i polinomi interpolanti Q e T quando vengano considerati approssimazioni delle funzioni.

Soluzione proposta

Il Programma proposto è costituito da 3 Function per l'elaborazione del Polinomio Interpolante Q e il Polinomio T, qualora per quest'ultimo fosse possibile, ovvero quando il Grado dell'Interpolazione N è Pari. Tale condizione è gestita alla fine della Function `f_interpolazione_linspacerand()` attraverso un Controllo IF sul Modulo Resto fra N e 2. Il programma sfrutta inoltre una nuova versione 2.0 della Function `funzione_atratti()`. In `funzione_tratti2()`.

```
%   Matematica Applicata e Computazionale
%
%   Interpolazione Line Space RAND
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
%
% USO
%
% Lanciare da consolle il file
%
% >> interpolazione_linspacerand
%
% e vedere i calcoli che esegue

%%%%%%%
% Main
%%%%%%%

uscita='No';
f_interpolazione_linspacerand(uscita);
```

```
%   Matematica Applicata e Computazionale
%
%   Interpolazione Line Space RAND
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function f_interpolazione_linspacerand.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NOTA: A DIFFERENZA DEI PRECEDENTI PROGRAMMI CHE PREVEDEVANO LA SELEZIONE
% DI UNA FUNZIONE O UNA SERIE DA STUDIARE, IN QUESTO CASO SI E' PREFERITO
% NON IMPLEMENTARE UNA FUNCTION APPOSITA CHE GESTISSE TALE SELEZIONE
% SVILUPPANDO IN TECNICA FUNCTION HANDLE: QUESTO PER UNA QUESTIONE DI
% SEMPLICITA' NELLA LETTURA DEL CODICE SORGENTE. DAL MOMENTO CHE
% L'ESERCIZIO PREVEDE, COME IL PRECEDENTE, LA POSSIBILITA' DI SELEZIONAR
% UNA FUNZIONE PERIODICA COMPOSTA
% x(pi-x) in [-pi,0]
% x(pi+x) in [0,+pi]

function f_interpolazione_linspacerand(uscita)

while strcmp(uscita,'No')
      % COSTRUZIONE MENU GRAFICO
      cellFunzioniP={'sin(2*x)';
      '(sin(8*x)).^2';
      'abs(x)';'x.^2';
      '1./1+x.^2';
      'sin(2*x)-sin(8*x)';
      'cos(22*x)';
      '(cos(8*pi*x)).^4';
```

```
'abs(sin(pi*x))'
'x.*(pi -/+ x)';};
kk=menu('Quale funzione',cellFunzioniP);
fString=char(cellFunzioniP{kk});

% FUNZIONE F COSTRUITA COME INLINE FUNCTION
F=inline(fString);

% ARRAY 2D (MATRICE) intervalli ABBINA ALLA FUNZIONE SCELTA
% I RELATIVI ESTREMI DEL DOMINIO IN FUNZIONE DEL PARAMETRO k
% CHE DETERMINA LA SCELTA DA MENU' SIA DELLA FUNZIONE CHE DEL SUO
% INTERVALLO
intervalli=[0,2*pi;
0,pi;
-1,1;
-1,1;
-5,5;
-pi,pi;
-1,1;
-2,2;
-1,1;
-pi,pi]; % ESSENDO L'INTERVALLO [-pi,0],[0,pi]

% DETERMINAZIONE ESTREMI INTERVALLO
% ATTRAVERSO INDICE RIGA CORRISPONDENTE ALLA FUNZIONE PERIODICA
% INDICE DI COLONNA CORRISPONDENTE ALL'ESTREMO DESTRO O SINISTRO
ja=1;
jb=2;
a=intervalli(kk,ja);
b=intervalli(kk,jb);

% SCELTA NUMERO NODI CON INPUT DIALOG BOX
titoloInputDialog='Dati Input';
nomeCampo='Numero Nodi (Np>0)';
default={'1'};
num_linee=1;
Np = inputdlg(nomeCampo,titoloInputDialog,num_linee,default);
Np = char(Np);
Np = str2num(Np);
N=Np-1; % DEFINISCE IL GRADO N IN BASE AL NUMERO DI NODI Np
```

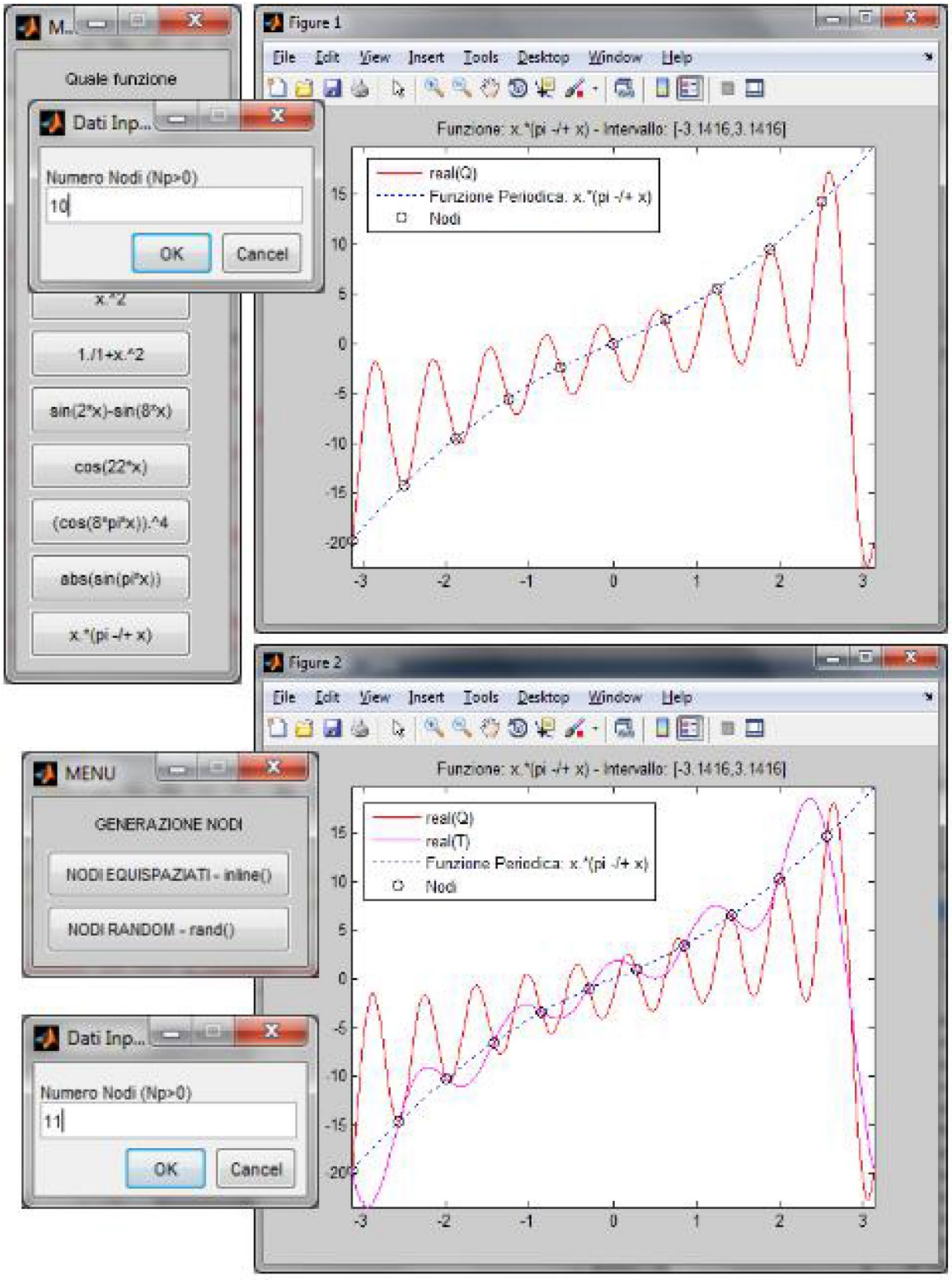
```
% MENU DI SCELTA TIPO DI NODI
% DETERMINA IN CHE MODO GENERARE IN NODI:
%  - CON linspace() [NODI EQUISPAZIATI]
%  - CON rand() [NODI RANDOM]
tipoNodi=menu('GENERAZIONE NODI',...
'NODI EQUISPAZIATI - linspace()','NODI RANDOM - rand()');

% COSTRUTTO SWITCH CASE - GESTISCE TIPOLOGIA NODI
switch tipoNodi

      case 1
            xi=linspace(a,b,Np+1)'; % Np+1 ASCISSE DI NODI
EQUISPAZIATI
            xi(end)=[]; % APERTURA INTERVALLO E SIZE Np

      case 2
            xi=a+((b-a)*rand(Np,1)); % Np RANDOM PROIETTATI AD
[a,b[

end % END SWITCH

x=linspace(a,b,199); % ASCISSE PER DISEGNARE FUNZIONE
z=exp(1i*x); % COMPONENTE ESPONENZIALE DI MATRICE OMEGA A

% CONTROLLO IF - ELSE
% CONTROLLA CHE LA SCELTA SIA O MENO CADUTA SULLA SELEZIONE
% DELLA FUNZIONE COMPOSTA CHE E' GESTITA DA UNA FUNCTION FATTA APPOSTA
if strcmp(fString,'x.*(pi -/+ x)')
      % ATTIVA LA FUNCTION FATTA APPOSTA PER CALCOLARE E DISEGNARE
      % UN POLINOMIO NELLO SPECIFICO CASO DELLA
      % FUNZIONE COMPOSTA x*(pi(-/+)x)
      [y,yi,c,c1,A] = funzione_atratti2(xi,x,Np);
else
      y=F(x); % ORDINATE DELLA FUNZIONE
      yi=F(xi); % ORDINATE DEI NODI
      k=0:N;
      A=exp(1i*xi*k); % MATRICE OMEGA
      c=A\yi; % SIS. LINEARE PER COEFFICIENTI c DI POLINOMIO Q

end % END IF
```

```
      % POLINOMI Q E T INTERPOLANTI - CREAZIONE E DISEGNO
      figure
      Q=polyval(flipud(c),z); %CALCOLO POLINOMIO Q

      % CONTROLLO IF
      % NEL CASO IL GRADO DEL POLINOMIO SIA PARI CREA ANCHE IL POLINOMIO T
      % ALTRIMENTI DISEGNA SOLO IL POLINOMIO Q GIA CREATO PER DEFAULT
      if mod(N,2)==0
            % COMPONENTE ESPONENZIALE CALCOLATA SU ASCISSE NODI
            zi=exp(1i*xi);
            c1=A\(yi.*zi.^(N/2));

            % SIS. LINEARE - COEFFICIENTI c1 DI POLINOMIO T
            T=z.^(-N/2).*polyval(flipud(c1),z); % CALCOLO POLINOMIO T
            plot(x,real(Q),'r',x,real(T),'m',x,y,'b:',xi,yi,'ok');

            % DISEGNA POLOMI Q, T, I NODI E LA FUNZIONE
            title(['Funzione: ' fString...
            ' - Intervallo: [' num2str(a) ',' num2str(b) ']']);
            legend(   'real(Q)',   'real(T)',['Funzione   Periodica:   '
fString],'Nodi');
            axis tight
      else
      % DISEGNA SOLO POLINOMIO Q, NODI E FUNZIONE
      plot(x,real(Q),'r',x,y,'b:',xi,yi,'ok');
      title(['Funzione: ' fString...
      ' - Intervallo: [' num2str(a) ',' num2str(b) ']']);
      legend( 'real(Q)', ['Funzione Periodica: ' fString],'Nodi');
      axis tight

      end % END IF

      % CRITERI DI SELEZIONE USCITA PER GESTIRE CICLO WHILE
      % ATTRAVERSO UNA QUESTION DIALOG BOX
      uscita=questdlg('Vuoi uscire?','Continua...');

end % END WHILE

end % END FUNCTION
```

```
%   Matematica Applica e Computazionale
%
%   Interpolazione Line Space RAND
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function funzione_atratti2.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [y,yi,c,c1,A] = funzione_atratti2(xi,x,Np)

N=Np-1;
fsx=@(t) t.*(pi-t); % FORMA DELLA FUNZIONE PER DOMINIO NEGATIVO
fdx=@(t) t.*(pi+t); % FORMA DELLA FUNZIONE PER DOMINIO POSITIVO

%%%% CICLO WHILE %%%%
% PER CALCOLARE LE ORDINATE DEI NODI DEL POLINOMIO INTERPOLANTE Q
% SI NECESSITA PER FORZA DI UNO CICLO ITERATIVO SULLE ASCISSE
% POICHE' QUESTE SONO CREATE SOLO ALLE VOLTE IN linspace(),
% MENTRE ALTRE VOLTE IN rand() E IN TAL CASO NON HANNO UN ORDINE PRECISO
% DA (-) A (+) QUINDI INDICIZZARE SIA LE COMPONENTI DI yi CHE LE xi
index=1;

while index<=Np
      % SI VALUTA LA FUNZIONE fsx IN OGNI ASCISSA SX tk NEGATIVA
      yi(index)=fsx(xi(index)<0);
      % SI VALUTA LA FUNZIONE fdx IN OGNI ASCISSA DX tk POSITIVA
      yi(index)=fdx(xi(index)>0);
      index=index+1;
end
```

```
%%%% CALCOLO DEI PUNTI DELLA FUNZIONE %%%%
% A DIFFERENZA DEI PUNTI DEI NODI CHE NON SONO ORDINATI, PER I PUNTI DELLA
% FUNZIONE POSSIAMO OMETTERE IL CICLO WHILE UTILIZZANDO LE PROPRIETA'
% BOOLEANE CHE SI POSSONO OPERARE SUGLI ARRAY.
y_sx=fsx(x(x<0)); % ORDINATE PER CAMPIONI SX tt NEGATIVI
y_dx=fdx(x(x>0)); % ORDINATE PER CAMPIONI DX tt POSITIVI
y= [y_sx';y_dx'];
k=0:N-1; % ARRAY DI ESPONENTI k

% ASCISSE PER LA FUNZIONE
A=exp(1i*xi*k) ; % MATRICE DEI VALORI ESPONENZIALI
c=A\yi; % COEFFICIENTI SOLUZIONE DEL SISTEMA LINEARE (POLINOMIO Q)
zi=exp(1i*xi); % VALORI ESPONENZIALI
c1=A\(yi.*zi.^(N/2));  % COEFFICIENTI SOLUZIONE DEL SISTEMA LINEARE
(POLINOMIO T)

end % END FUNCTION
```

Esempio d'uso



Fourier

Es. 39 – Liv. 2 – Stringa scorrevole

Mediante `circshift()` simulare, in una finestra grafica, una stringa animata che scorre da destra a sinistra.

Soluzione proposta

L'elaborato consiste in uno Script che ha funzione di assunzione Input della Stringa da animare, di default è assegnato proprio il Titolo completo della Traccia di Esercizio.

Passando tale input alla Function `f_ruotastringa()`, implementata per essere il cuore del programma, si attua lo Scorrimento a Video della Stringa assegnata.

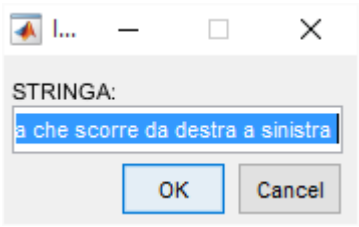
La Stringa viene passata in input alla Matlab Function circshift(), che vede un qualsiasi Array come Colonna e provoca lo Spostamento di tutte le componenti di j Posizioni definite da input come la Direzione (+/-), quindi verso l'Alto (+j) o verso il Basso (-j), ma quando una componente Eccede Oltre l'Estremo dell'Array, circshift lo riposiziona all'Estremità Opposta e lo scorrimento continua. Possiamo rendere orizzonatale l'Array Colonna per Trasposizione (') e simulare così uno Scorrimento Orizzontale, sequenzializzandolo attraverso una serie di Visualizzazioni plot() Successive poiché si indicizza sulle componenti della nostra Stringa un Ciclo While Infinito, che termina grazie al PushBotton definitio appositamente con la Function `uicontrol()` per Interfaccia GUI.

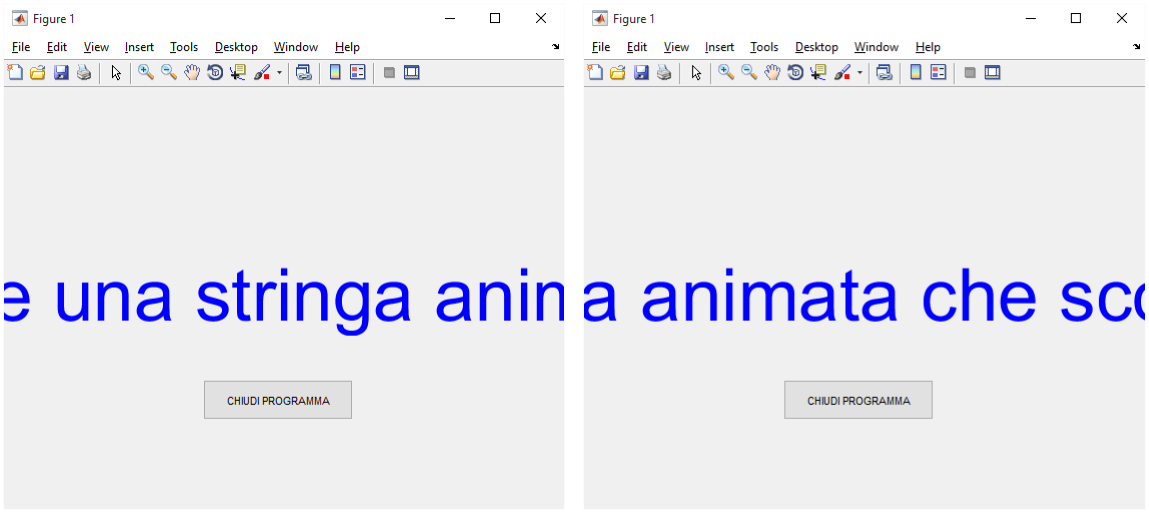
```
%   Matematica Applica e Computazionale
%
%   Interpolazione Line Space RAND
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
%
% USO
%
% Lanciare da consolle il file
%
% >> stringascorrevole
%
% e vedere i calcoli che esegue

%%%%%%%
% Main
%%%%%%%
% MENU' DI INSERIMENTO STRINGA DA RUOTARE

campi={'STRINGA: '};
titoloDLG='Inserisci Stringa da Ruotare';
numeroRighe=1;
valoriDefault={'Esercizio N°39 [liv. 2] - Mediante circshift() simulare
una stringa animata che scorre da destra a sinistra '};
datiInput = inputdlg(campi,titoloDLG,numeroRighe,valoriDefault);
datiInput = char(datiInput);
stringa=datiInput;

f_scorristringa(stringa); % carica stringa

close all
```

```
%   Matematica Applica e Computazionale
%
%   Interpolazione Line Space RAND
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function f_scorristringa.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function f_scorristringa(stringa)

% USER INTERFACE PER BOTTONE DI CHIUSURA PROGRAMMA
uicontrol('Style','Pushbutton',...
'Position',[200 90 150 40],...
'String','CHIUDI PROGRAMMA',...
'Callback','close');
% CICLO WHILE (INFINITO)
% SERVE A SPOSTARE DI VOLTA IN VOLTA I CARATTERE DELLA STRINGA VERSO CELLE
% DELL'ARRAY DI POSIZIONI SUCCESSIVE, SIMULANDO COSI' IL MOVIMENTO DELLA
% STRINGA
j=0;
infinito=1;

while infinito==1
      % SPOSTA A DX(-) I CARATTERI DI stringa DI j POSIZIONI OGNI VOLTA
      c=(circshift(stringa',-j))';
      h=text(.5,.5,c,'FontSize',55,'HorizontalAlignment','center','color'
,'b');
      axis off; % ELIMINA GLI ASSI
      pause(.25) % PAUSA 1/4 DI SECONDO
```

```
        j=j+1;  % INCREMENTA INDICE PER SPOSTARE COMPONENTI A POSIZIONE
SUCCESSIVA
        set(h,'Visible','off') % ELIMINA L'ULTIMA VISUALIZZAZIONE

end % END WHILE

end % END Function
```

Esempio d'uso





Che si ripete all'infinito finché non si chiude il programma

Es. 41 – Quiz DFT

Quale relazione lega i coefficienti del polinomio trigonometrico interpolante ed una DFT?

Soluzione proposta

Sappiamo per Teorema, che Assegnati N+1 Nodi, zk (xk , yk) per k=0,1,…N, esiste un unico Polinomio Trigonometrico del tipo Q(x)=c0 + c1eix + c2e2ix + … + cNeNix *tale che* Q(xk)=yk.

E i coefficienti di tale Polinomio Q sono dati dalla soluzioni c del seguente Sistema Lineare

(Sistema Lineare) W*c=y → c=W/y (Coefficienti)

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_{N+1} & \omega_{N+1}^{2} & \cdots & \omega_{N+1}^{N} \\ 1 & \omega_{N+1}^{2} & \omega_{N+1}^{4} & \cdots & \omega_{N+1}^{2N} \\ 1 & \omega_{N+1}^{3} & \omega_{N+1}^{6} & \cdots & \omega_{N+1}^{3N} \\ \vdots & & \vdots & \cdots & \vdots \\ 1 & \omega_{N+1}^{N} & \omega_{N+1}^{2N} & \cdots & \omega_{N+1}^{N^2} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} \quad \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{i\frac{2\pi}{N+1}} & e^{i\frac{4\pi}{N+1}} & \cdots & e^{i\frac{2\pi N}{N+1}} \\ 1 & e^{i\frac{4\pi}{N+1}} & e^{i\frac{8\pi}{N+1}} & \cdots & e^{i\frac{4\pi N}{N+1}} \\ \vdots & & \vdots & \cdots & \vdots \\ 1 & e^{i\frac{2\pi N}{N+1}} & e^{i\frac{4\pi N}{N+1}} & \cdots & e^{i\frac{2\pi N^2}{N+1}} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

Dove la Matrice W è una particola Matrice di Valori Esponenziali $W=e^{i2\pi/N+1}$ elevati a potenza del prodotto tra indice di Riga e Colonna k*h tali valori sono proprio le Radici n-sime dell'Unità Complesse È da notare la Simmetria delle Componenti rispetto la Diagonale Principale.

$$W_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

Per le proprietà della Matrice W possiamo però affermare che essendo c=W\y allora possiamo calcolare i Coefficienti c allo stesso modo attraverso la sua Matrice Inversa $W^{-1}= W'/N$ à $c=W^{-1}*y$, ma poiché W è Simmetrica rispetto la Diagonale Principale, la Trasposta Hermitiana W' coincide con la Complessa Coniugata conj(W) quindi i Coefficienti c possono essere calcolati direttamente

come come c=(W'/N)*y

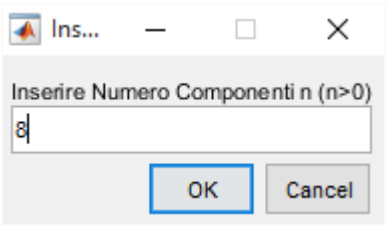
Nel contempo, una DFT (Discrete Fourier Transform) può essere definita in Generale come una Funzione F risultato di una Trasformazione di un'altra Funzione f, e come sappiamo, in Geometria una Trasformazione si ottiene tramite il Prodotto con una Matrice di Trasformazione, che nel caso della DFT è la Matrice Ω tale che F= Ω*f

```
%   Matematica Applicata e Computazionale
%
%   Interpolazione Line Space RAND
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
%
% USO
%
% Lanciare da consolle il file
%
% >> quizdft
%
% e vedere i calcoli che esegue

%%%%%%%
% Main
%%%%%%%

% QUIZ: Quale relazione lega i coefficienti del polinomio trigonometrico
% interpolante ed una DFT?
uscita='No';
while strcmp(uscita,'No');

      clc % PULISCI SCHERMO MATLAB

      % MENU' DI SCELTA DIMENSIONE DEL SEGNALE
      % NUMERO DI COMPONENTI DELLA FUNZIONE VALUTATA
      campi={'Inserire Numero Componenti n (n>0)'};
      titoloDLG='Inserire Numero Componenti';
      numeroRighe=1;
      valoriDefault={'1'};
```

```
    datiInput = inputdlg(campi,titoloDLG,numeroRighe,valoriDefault);
    datiInput = char(datiInput);
    datiInput = str2num(datiInput);
    n=datiInput;
    k=0:n-1;
    hk=k'*k;
    x=linspace(0,2*pi,n+1); x(end)=[];
    y=(sin(x)+cos(x))';

    % W E' LA MATRICE OMEGA (W)
    % CHE HA PER COMPONENTI LE RADICI PRIMITIVE DELL'UNITA N-SIMA
    % E RAPPRESENTA LA MATRICE DI TRASFORMAZIONE DEL SISTEMA LINEARE
    % W*c=y CON CUI SI POSSONO CALCOLARE I COEFFICIENTI c DEL POLINOMIO
    % INTERPOLANTE LA FUNZIONE y IN n NODI EQUISAPZIATI
    w=exp(1i*2*pi/n); % RADICE n-ESIMA PRIMITIVA DELL'UNITA'
    W=w.^rem(hk,n); % MATRICE ESPONENZIALE PER NODI EQUISPAZIATI

    % LA RELAZIONE CHE LEGA LA MATRICE OMEGA W AD UNA DFT RISIEDE NEL
    % FATTO CHE LA MATRICE W, OLTRE A GARANTIRE IL CALCOLO DEI
    % COEFFICIENTI DI UN POLINOMIO INTERPOLANTE SU NODI EQUISPAZIATI, E'
    % TALE CHE LA SUA COMPLESSA CONIUGATA conj(W) COSTITUISCE PROPRIO LA
    % MATRICE UTILIZZATA DALLA TRASFORMATA DISCRETA DI FOURIER DFT
    %%% PROVA LEGAME %%%%
    % STESSO RISULTATO SE SI CALCOLANO I COEFFICIENTI DEL POLINOMIO Q
    % TRAMITE FFT OPPURE CON CALCOLO SULLA COMPLESSA CONIUGATA DI W
    % (O ANCHE CON INVERSA OPPURE c=inv(W)*y. POICHE' W SIMMETRICO)
    c1=(conj(W)/n)*y;

    % CALCOLO DEI COEFFICIENTI TRAMITE DFT CALCOLATA CON ALGORITMO FFT
    c2=fft(y)/n;

    display('Coefficienti c del Polinomio Interpolante Q(x)');
    display(' ');
    display('ESEMPIO: Funzione y=sin(x)+cos(x)');
    display('Calcolo (W^-1)*y - Calcolo fft()');
    display([c1 c2]);
    disp(' ');
    disp('STESSI VALORI X ENTRAMBI GLI ALGORITMI! LEGAME DIMOSTRATO');
    uscita=questdlg('Vuoi uscire?','USCITA');
end
```

```
% ATTENZIONE!
% SI 'E PREFERITO USARE display() PER UNA VISUALIZZAZIONE A COMMAND WINDOWS
% POICHE CON UNA FUNCTION text PER VISUALIZZARE RISULTATI A FINESTRA
% GRAFICA, IN QUANTO UNA FUNCTION text()SEMBRA ALTERARE LA FORMA VALORI
% COMPLESSI CHE RAPPRESENTA COME REALI IN FORMA ESPONENZIALE!! (????)
% CHE AD OGNI MODO E' UNA NOTAZIONE SCOMODA
% DA OSSERVARE PER EFFETTUARE VALUTAZIONI DEL NOSTRO TIPO.
% IN CASO CONTRARIO SI SAREBBE POTUTO UTILIZZARE UN CODICE COME QUELLO
% SOTTOSTANTE

%%% MODALITA' TEXT WINDOWS - text() %%%
% LIBERA COMMENTI SOTTOSTANTI
%{
figure
text(.5,.5,num2str(c1),'FontSize',12,...
'HorizontalAlignment','center','Color','b','FontWeight','bold');
axis off
figure
text(.5,.5,num2str(c2),'FontSize',12,...
'HorizontalAlignment','center','Color','b','FontWeight','bold');
axis off
%}
```

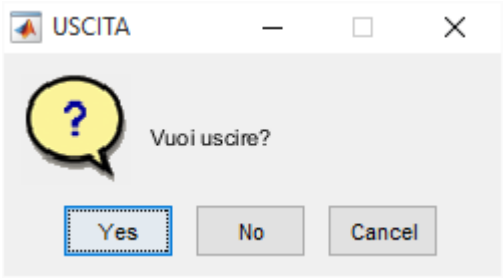
Esempio d'uso




```
Coefficienti c del Polinomio Interpolante Q(x)

ESEMPIO: Funzione y=sin(x)+cos(x)
Calcolo (W^-1)*y - Calcolo fft()

ans =

  -0.0000 + 0.0000i  -0.0000 + 0.0000i
   0.5000 - 0.5000i   0.5000 - 0.5000i
   0.0000 - 0.0000i   0.0000 + 0.0000i
   0.0000 - 0.0000i   0.0000 - 0.0000i
```

```
   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 - 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.5000 + 0.5000i   0.5000 + 0.5000i
```

STESSI VALORI X ENTRAMBI GLI ALGORITMI! LEGAME DIMOSTRATO



Es. 47 – Liv. 1 – Approssimazione Fourier

Approssimare numericamente nell'intervallo [-T/2,+T/2] alcune funzioni $f(x)$, periodiche e non, mediante ridotta di ordine N della sua Serie di Fourier costruendo i coefficienti di Fourier sia mediante integrazione numerica (function quad()) sia mediante DFT (function `fft()` e `fftshift()`) e confrontare i risultati ottenuti al variare di T ed N.

Soluzione proposta

L'elaborato implementa i 2 Algoritmi per il calcolo dei Coefficienti di Fourier c utilizzati per elaborare la Ridotta Sn della Serie di Fourier di una Funzione, tale che la Ridottada possa Approsimare la Funzione. Dopo un Input Dati per Interfaccia Grafica che prevede l'inserimento della Funzione da Approssimare fun, la Grandezza di Intevallo T in cui valutare, l'Ordine N della Ridotta e l'Ordine Massimo Nmax della Rispettiva Serie di Fourier, il programma Chiede da Menu Grafico a Bottoni, quela sia il tipo di Algoritmo da Utilizzare per Creare la Funzione Ricostruita e Disegnarla assieme alla Funzione da Approssimare; un Costrutto SWITCH CASE dirige l'esecuzione sulla porzione di codice d'interesse alla scelta.

Al momento del Disegno, per ogni Livello D'Ordine N della Ridotta della Funzione Ricostruita, viene scelto un Colore Diverso per la Curva, tramite una serie di Controlli IF…IFELSE sul plot(), dal Colore Piu' Scuro (Blu) a Colore Piu' Chiaro (Giallo) esistono 4 Livelli di Percentuali, tanto Piu' Alta è la Percentuale, tanto piu' Chiaro è il Colore della Curva e tanto Maggiore è l'Ordine della Ridotta N:

GIALLO: Ridotte di Ordine N compreso fra 80 e Nmax (N>80%)

CIANO : Ridotte di Ordine N compreso fra 60 e 80 (N>60%)

VERDE : Ridotte di Ordine N compreso fra 20 e 60 (N>20%)

BLU : Ridotte di Ordine N inferiore a 20 (N<20%)

Inolte, nel caso di Algoritmo di Quadratura, la Ridotta Sn viene creata con un'indicizzazione sulla Matrice Coefficienti c del tipo c(m+N/2:-1:m-N/2) ovvero valutando la polyval() sui coefficienti al contrario, dal Minore al Maggiore, Disribuendo il Periodo T attorno al Punto Medio m. Mentre nel caso di Algoritmo DFT utilizziamo la Matrice Coefficienti Invertita flipud(c) rispetto ai Triangoli Superiore e Inferiore, per calcolare con polyval() il Polinomio Interpolante che costruisce la Ridotta Sn.

E' stato inoltre gestito il caso del fattore (-1)k k=-N/2…N/2 attraverso un Controllo IF..ELSE sulla Semiampiezza dell'Intervallo, che vale sempiIntervallo=N/2, valutando il caso in cui sia

Pari: Parte da 2a Componente di C.F. [c(2:2:end) = -c(2:2:end)]

Dispari: Parte da 1a Componente di C.F. [c(1:2:end) = -c(1:2:end)]

Nel caso di N/2 Pari si cominciano a rendere Negativi i Coefficienti di Fourier a Coppie Alterne partendo dalla 2a Componente, mentre nel caso di

N/2 Dispari si rendono Negative le Coppie Alterne a partire dalla 1a Componente Gli Output sono frutto del test di varie funzioni, alcune di esse oggetto d'esempi del corso. Il Test sulla Funzione sign() (Segno) è eseguita con un'Integrazione Numerica quad(), ed è un esempio di come tale Algoritmo si dimostri Instabile, quindi Inefficiente, all'Aumentare dell'Ordine della Ridotta. Crescendo l'Ordine, l'Interpolazione perde via via di precisione.

Nel caso della Funzione sin(6*x), invece, la cui Approssimazione è operata con Algoritmo DFT, dimostra come questo sia un Algoritmo Efficiente, poiché piu' all'Aumentare dell'Ordine della Ridotta, Aumenta la Precisione dell'Approssimazione. Caso particolare N=Nmax la Curva Approssimante risulta Precisamente Uguale alla Funzione da Ricostruire, eccetto sulle Estremità della Curva, dove oscilla un minimo. Verifica questa tesi anche il Test che sulla Funzione sqrt(x/3) con Algoritmo di Integrazione Numerica

```
%
%   Matematica Applicata e Computazionale
%
%   Approssimazione di Fourier
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
%
% USO
%
% Lanciare da consolle il file
%
% >> approssfourier
%
% e vedere i calcoli che esegue

%%%%%%%
% Main
%%%%%%%

uscita='No';
f_appross_seriefourier(uscita);
```

```
%
%   Matematica Applicata e Computazionale
%
%   Approssimazione di Fourier
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function f_approssfourier.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function f_appross_seriefourier(uscita)
while strcmp(uscita,'No')
      % MENU' DI SCELTA DELLE FUNZIONI
      [fun , fString] = scelta_funzioneSF();

      % MENU' DI SCELTA DELLE COMPONENTI CARTESIANE [a,b] E DELL'ORDINE k
      campi={'Periodo T','Ordine Ridotta (N<=Nmax)',...
      'Ordine Masimmo Nmax'};
      titoloDLG='Input di Dati';
      numeroRighe=1;
      valoriDefault={'1','1','2'};
      datiInput = inputdlg(campi,titoloDLG,numeroRighe,valoriDefault);
      datiInput = char(datiInput);
      T=str2num(datiInput(1,:)); % GRANDEZZA INTERVALLO
      N=str2num(datiInput(2,:)); % ORDINE RIDOTTA Sn DELLA SERIE DI FOURIER
      Nmax=str2num(datiInput(3,:)); % ORDINE MASSIMO DELLA RIDOTTA Sn

      % MENU' DI SCELTA ALGORITMO
      algoritmo=menu('Scegli l''Algoritmo','Integrazione - quad()',...
      'DFT - fft() fftshift()');
```

```
x=linspace(-T/2,T/2,500);
y=feval(fun,x); % VALUTA LA FUNZIONE NEL DOMINIO (ORDINATE fun)

% COSTRUTTO SWITCH CASE - SCELTA ALGORITMO
switch algoritmo

% ALGORITMO INTEGRAZIONE NUMERICA quad()
case 1
      % CONCATENZIONE PER DEFINIRE fun IN MODO COMPLESSO
      argomento=[fString '.*exp(-i*2*k*pi/T*x)'];

      % CREA CORRISPETTIVA FUNZIONE COMPLESSA DI fString (O fun)
      qfun=inline(argomento, 'x','T','k');
      c=[]; % INIZIALIZZA COEFFICIENTI DI FOURIER

      % INTEGRAZIONE NUMERICA DELLA FUNZIONE COMPLESSA SUL DOMINIO
      % DEFINITO CON DIMENSIONI PARI ALL'INTERVALLO DESIDERATO T
      for k=-Nmax/2:Nmax/2
            % Q PER I VALORI DI INTEGRAZIONE
            % fcnt E' UN CONTATORE DI VALUTAZONi DI quad() (NON USATO)
            [Q,fcnt] = quad(qfun,-T/2,T/2,[ ],[ ],T,k);
            c=[c;Q/T]; %AD OGNI PASSO, CONCATENAZIONE IN COLONNA DEI
            % RISULTATI DELL'INTEGRAZIONE IN RAPPORTO AL
            % PERIODO
      end

      m=Nmax/2+1; % INDICE MEDIO SU COEFFICIENTI DI FOURIER
      Sn=exp(-1i*N*pi/T*x).*polyval(c(m+N/2:-1:m-
N/2),exp(1i*2*pi/T*x));

% ALGORITMO DFT fft() fftshift()
case 2
      xj=T/N.*(-N/2:N/2)'; % ASCISSE DEI CAMPIONI
      fj=feval(fun,xj); % CALCOLO ORDINATE DEI CAMPIONI
      % PARTICOLARE MATRICE f DOVE LA PRIMA RIGA E' LA MEDIA DEI
      % VALORI DEI CAMPIONI AGLI ESTREMI, MENTRE LA SECONDA RIGA
      % E' COSTITUITA DAI VALORI FRA ESSI INTERMEDI
      f=[.5*(fj(1)+fj(end));fj(2:end-1)];

      % APPLICO SU TALE MATRICE L'ALGORITMO fft() E INVERTO CON
      % fftshift() LE DUE META' DELL'ARRAY RISULTATO
      c=fftshift(fft(f));
```

```
        % NEI COEFFICIENTI DI FOURIER c IL PRIMO DEVE ESSERE UGUALE
        % ALL'ULTIMO, QUINDI LO AGGIUNGO IN CODA E SCALO TUTTI I
        % COEFFICIENTI DI UN FATTORE Nmax
        c=[c; c(1)]/Nmax;

        % CONTROLLO IF - COEFFICIENTI NEGATIVI E POSITIVI
        % STABILISCE A PARTIRE DA QUALE COEFFICIENTE BISOGNA
        % COMINCIARE A CAMBIARE IL SEGNO ALLE COPPIE ALTERNE
        % STABILISCO QUESTO CONTROLLANDO SE LA SEMIAMPIEZZA E' PARI O
        % DISPARI
        semiIntervallo=N/2;
        if mod(semiIntervallo,2)==0 % SE IL SEMINTERVALLO E' PARI
              c(2:2:end) = -c(2:2:end); % PARTI DA 2° COMPONENTE
        else % ALTRIMENTI
              c(1:2:end) = -c(1:2:end); % PARTI DALLA PRIMA
        end

        % INVERSIONE TRIANGOLARE SUPERIORE / INFERIORE
        Sn=exp(-1i*N*pi/T*x).*polyval(flipud(c),exp(1i*2*pi/T*x));

end % END SWITCH

% PERCENTUALE RIDOTTE RISPETTO L'ORDINE MAX
% CALCOLA A QUALE RANGE DI RIDOTTE APPARTIENE LA CURVA DELL'ORDINE N
% CHE SI E' SELEZIONATO. QUANTO PIU' GRANDE E' L'ORDINE DELLA RIDOTTA
% TANTO PIU' ALTA E' LA PERCENTUALE, TANTO PIU' CHIARO IL COLORE
% DELLA CURVA DELLA RIDOTTA
percentuale=(N*100)/Nmax;
hold on

if percentuale>=81 % SE RIDOTTA DI ORDINE MOLTO ALTO 80%-100%
      valoreString='Ordine > 80%';
      color='y:';
elseif percentuale>=61 % SE RIDOTTA E' DEL RANGE 60%-80%
      valoreString='Ordine > 60%';
      color='c:';
elseif percentuale>=21 % SE RIDOTTA DI ORDINE MEDIO/BASSO 20%-60%
      valoreString='Ordine > 20%';
      color='g:';
elseif 1<=percentuale<=20 % SE RIDOTTA DI ORDINE BASSO 0%-20%
      valoreString='Ordine < 20%';
```

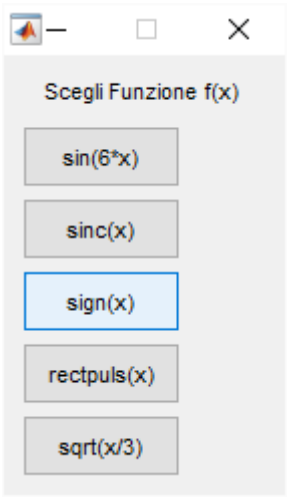
```
            color='b:';

        end % END IF-ELSE

        % DISEGNA FUNZIONE IN NERO E CURVA D'APPROSSIMAZIONE DI VARIO COLORE
        plot(x,y,'k',x,real(Sn),color,'LineWidth',2)

        % CREA STRINGA PER TITOLO GRAFICO
        titolo=['Ridotta di f(x) = ' fString ' - Ordine Ridotta N = ',...
        num2str(N) ' - ' valoreString];

        % IMPOSTA STRINGA COME TITOLO
        title(titolo);
        axis tight

        % CRITERI DI SELEZIONE USCITA PER GESTIRE CICLO WHILE
        % ATTRAVERSO UNA QUESTION DIALOG BOX
        uscita=questdlg('Vuoi uscire?','Continua...');

end % END WHILE

end % END FUNCTION
```

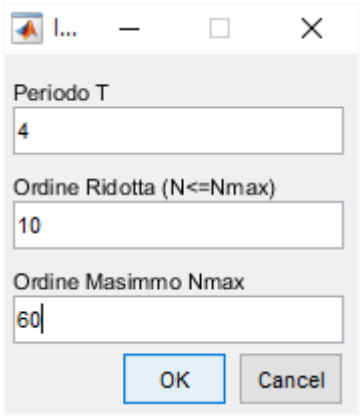
```
%   Matematica Applicata e Computazionale
%
%   Approssimazione di Fourier
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function scelta_funzioneSF.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [f,fString] = scelta_funzioneSF()
% FUNZIONI TEST
cellFunzioni={'sin(6*x)';
'sinc(x)';
'sign(x)';
'rectpuls(x)'
'sqrt(x/3)'};

funzioneScelta = menu('Scegli Funzione f(x)',cellFunzioni);
switch funzioneScelta
      case 1
            % SENO DI 6x f(x)=sin(6*x)
            f=@(x) sin(6*x);

      case 2
            % RAPPORTO SENO E SUO ARGOMENTO
            f=@sinc; %sin(t)/t

      case 3
            % SEGNO f(x)=sign(x)
            f=@sign;
```

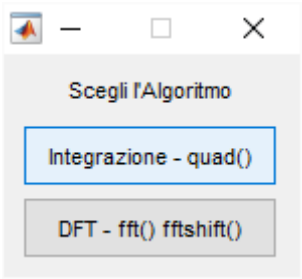
```
        case 4
            % IMPULSO RETTANGOLARE f(t)=rectpuls(t)
            f=@rectpuls;

        case 5
            f=@(x) sqrt(x/3);

        end % END SWITCH

        % STRINGA CON NOME FUNZIONE
        fString=cellFunzioni{funzioneScelta};
end % end function
```

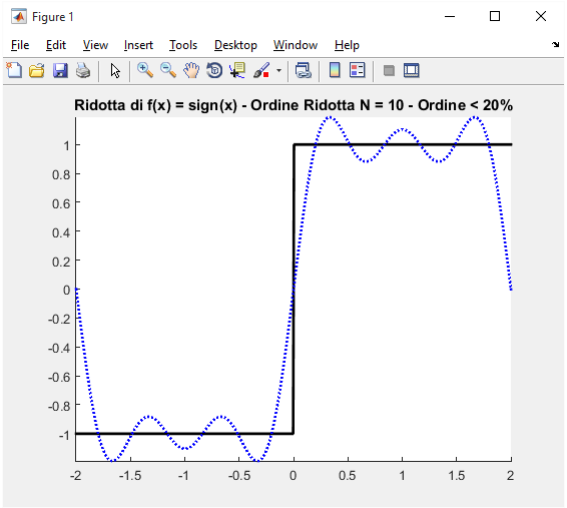
Esempio d'uso

$f(x) = \text{sign}(x)$ non periodica

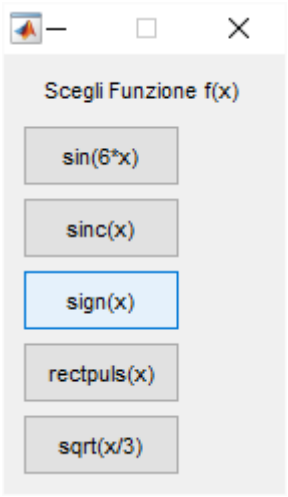


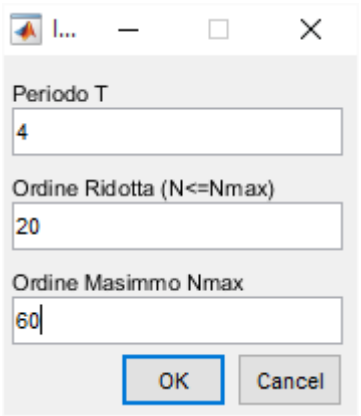


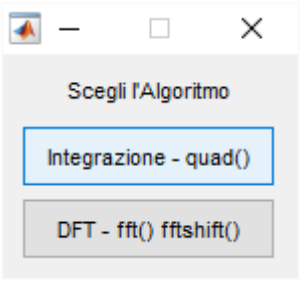


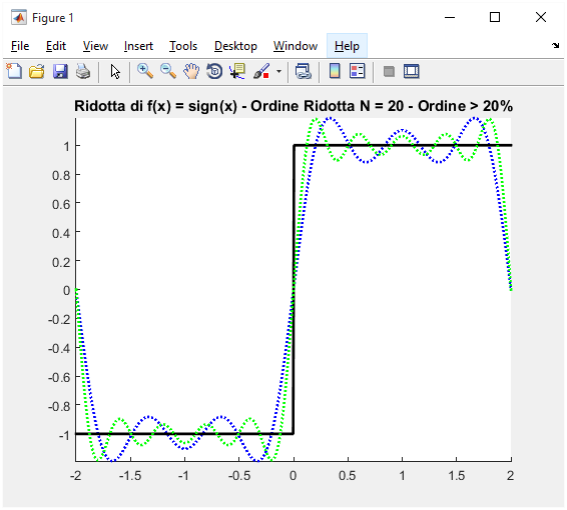


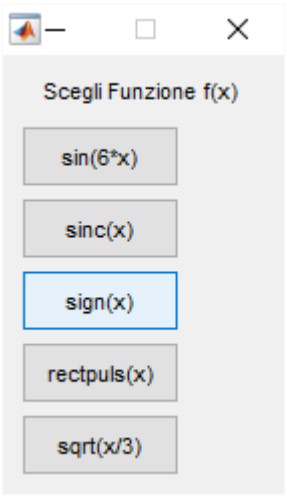
Vediamo sulla stessa figura come si comporta

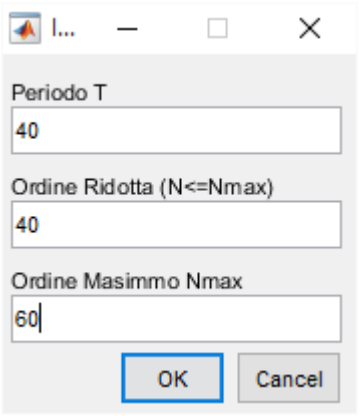


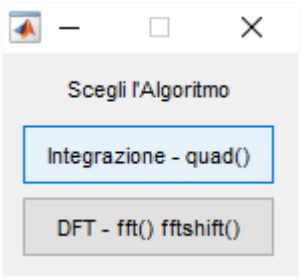







Scegli Funzione f(x)
sin(6*x)
sinc(x)
sign(x)
rectpuls(x)
sqrt(x/3)


I...
Periodo T
40
Ordine Ridotta (N<=Nmax)
40
Ordine Masimmo Nmax
60
OK
Cancel


Scegli l'Algoritmo
Integrazione - quad()
DFT - fft() fftshift()

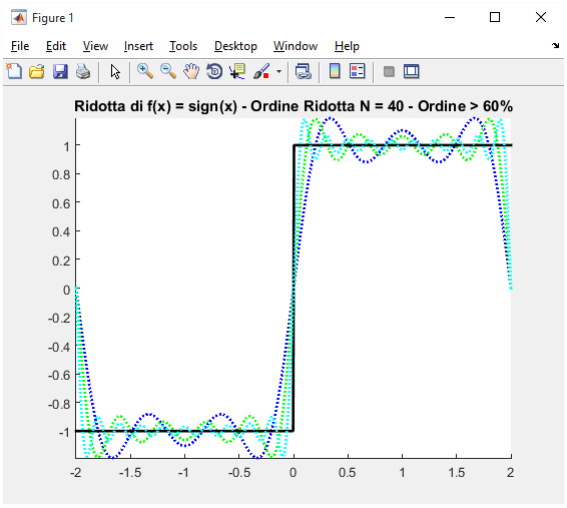
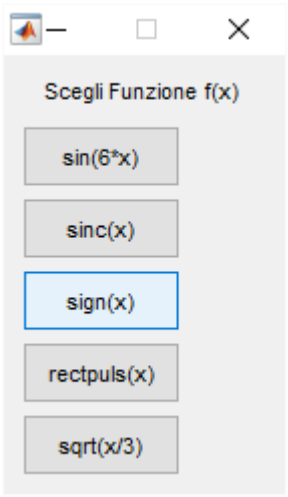
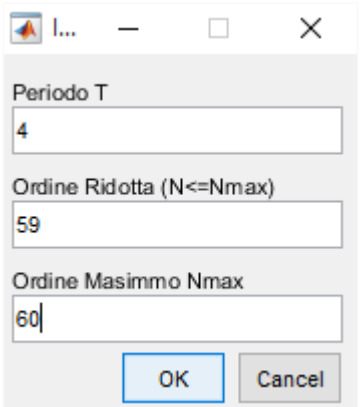
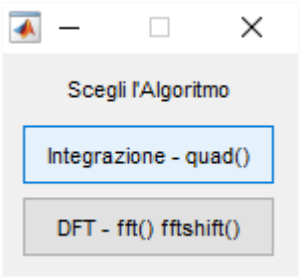


Figure 1
File Edit View Insert Tools Desktop Window Help
Ridotta di f(x) = sign(x) - Ordine Ridotta N = 40 - Ordine > 60%
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Scegli Funzione f(x)
sin(6*x)
sinc(x)
sign(x)
rectpuls(x)
sqrt(x/3)


I...
Periodo T
4
Ordine Ridotta (N<=Nmax)
59
Ordine Masimmo Nmax
60
OK
Cancel


Scegli l'Algoritmo
Integrazione - quad()
DFT - fft() fftshift()

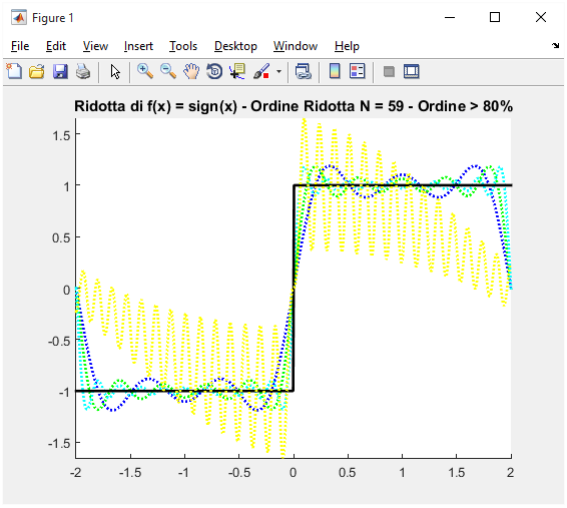
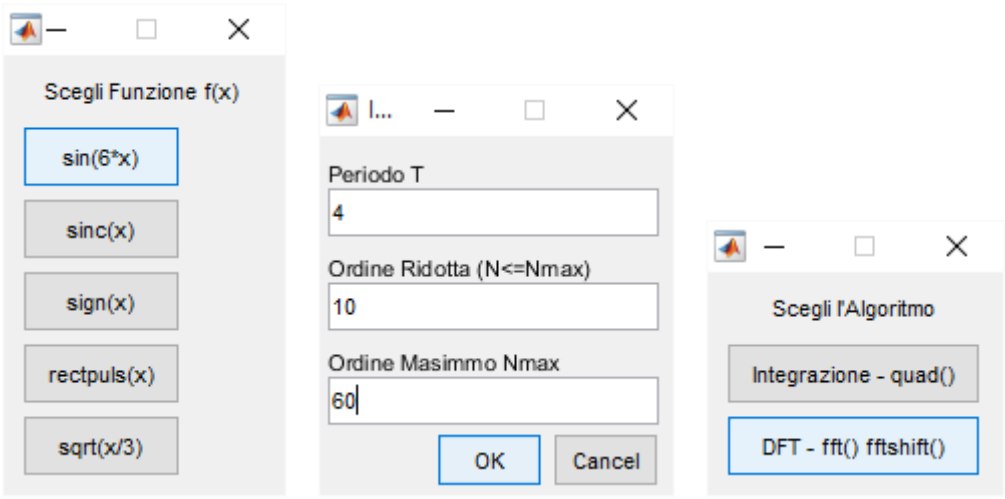
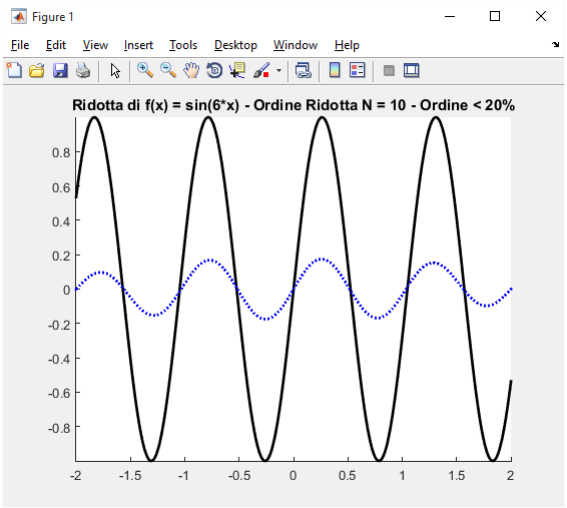
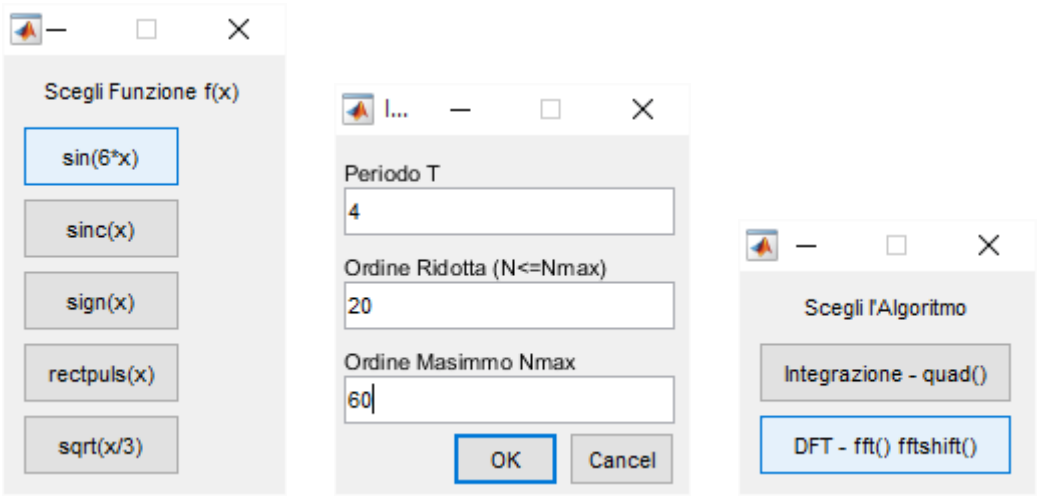


Figure 1
File Edit View Insert Tools Desktop Window Help
Ridotta di f(x) = sign(x) - Ordine Ridotta N = 59 - Ordine > 80%
1.5
1
0.5
0
-0.5
-1
-1.5
-2 -1.5 -1 -0.5 0 0.5 1 1.5 2

Consideriamo ora la funzione periodica sin(6*x) con gli stessi valori inseriti in precedenza






Scegli Funzione f(x)
sin(6*x)
sinc(x)
sign(x)
rectpuls(x)
sqrt(x/3)
I...
Periodo T
4
Ordine Ridotta (N<=Nmax)
20
Ordine Masimmo Nmax
60
OK
Cancel
Scegli l'Algoritmo
Integrazione - quad()
DFT - fft() fftshift()

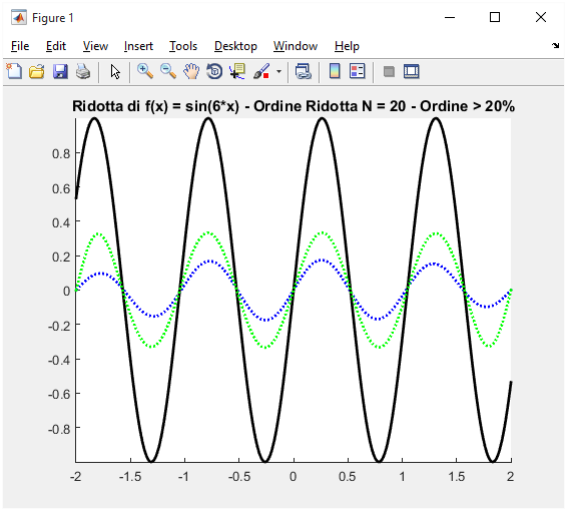
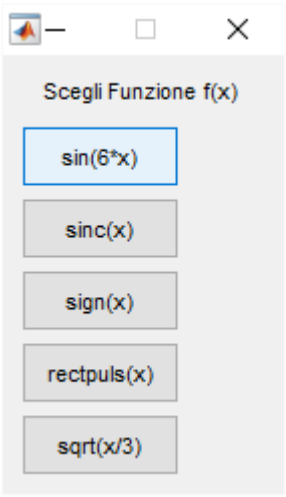
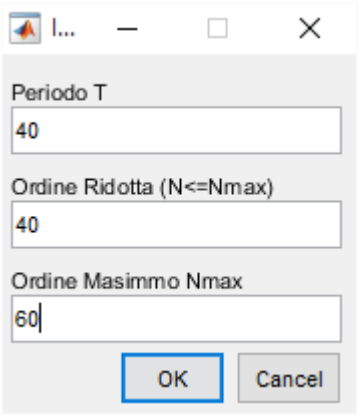
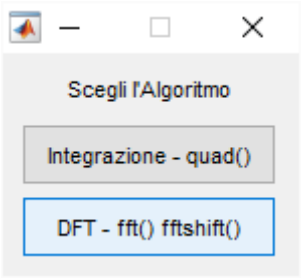


Figure 1
File Edit View Insert Tools Desktop Window Help
Ridotta di f(x) = sin(6*x) - Ordine Ridotta N = 20 - Ordine > 20%
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Scegli Funzione f(x)
sin(6*x)
sinc(x)
sign(x)
rectpuls(x)
sqrt(x/3)


I...
Periodo T
40
Ordine Ridotta (N<=Nmax)
40
Ordine Masimmo Nmax
60
OK
Cancel


Scegli l'Algoritmo
Integrazione - quad()
DFT - fft() fftshift()

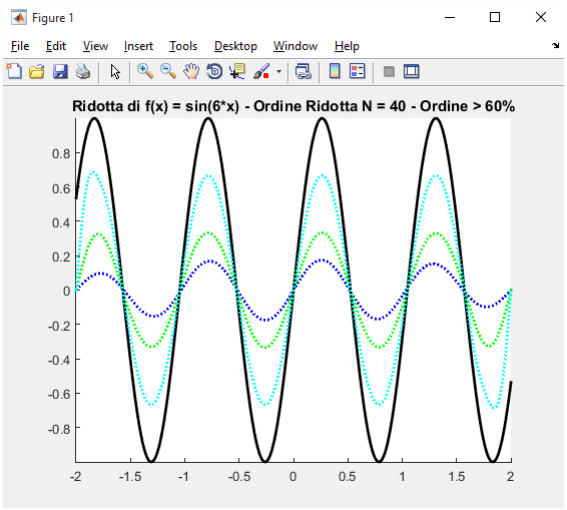
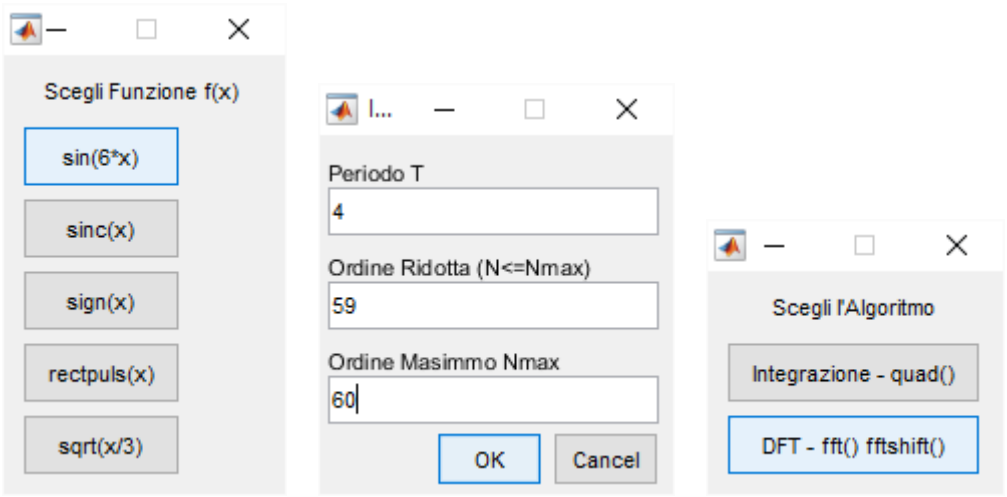


Figure 1
File Edit View Insert Tools Desktop Window Help
Ridotta di f(x) = sin(6*x) - Ordine Ridotta N = 40 - Ordine > 60%
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Scegli Funzione f(x)
sin(6*x)
sinc(x)
sign(x)
rectpuls(x)
sqrt(x/3)
I...
Periodo T
4
Ordine Ridotta (N<=Nmax)
59
Ordine Masimmo Nmax
60
OK
Cancel
Scegli l'Algoritmo
Integrazione - quad()
DFT - fft() fftshift()

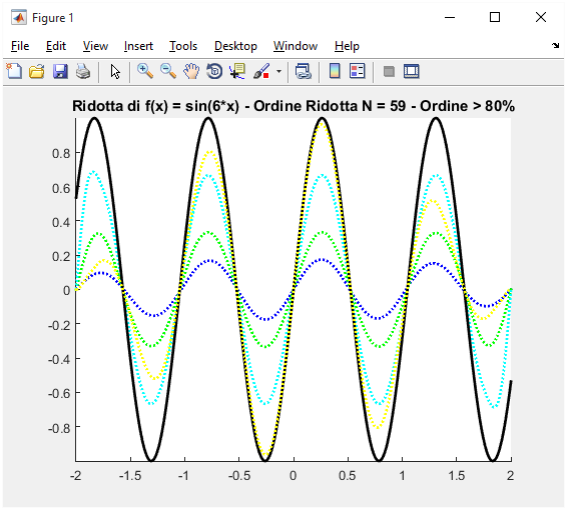


Figure 1
File Edit View Insert Tools Desktop Window Help
Ridotta di f(x) = sin(6*x) - Ordine Ridotta N = 59 - Ordine > 80%
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

Es. 51 – Liv. 1 – Approssimazione Fourier

Approssimare numericamente, in un intervallo [-T/2,T/2], la *Trasformata di Fourier* di alcune funzioni $f(x)$ di cui sia nota la trasformata analitica. Visualizzare (sia rispetto alla frequenza angolare che a quella circolare) e commentare i risultati ottenuti al variare dei parametri di discretizzazione T e N confrontandoli con la trasformata analitica

```
%   Matematica Applicata e Computazionale
%
%   Approssimazione Trasformaata di Fourier
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%
%
% USO
%
% Lanciare da consolle il file
%
% >> appross_trasfourier
%
% e vedere i calcoli che esegue

%%%%%%%
% Main
%%%%%%%

uscita='No';
f_appross_trasfourier(uscita);
```

```
%   Matematica Applicata e Computazionale
%
%   Approssimazione Trasformaata di Fourier
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function f_appross_trasfourier.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function f_appross_trasfourier(uscita)

% VARIABILI SYMBOLIC PER CALCOLO DELLA TRAFORMATA FOURIER ANALITICA
syms t v w real

while strcmp(uscita,'No')
      % SCELTA FUNZIONE
      [fun, Fv, fString] = scelta_funzioneFT(t,v,w);

      % MENU' DI SCELTA DELLE COMPONENTI CARTESIANE [a,b] E DELL'ORDINE k
      campi={'Ampiezza T','Numero Campioni N (N<0)'};
      titoloDLG='Input di Dati';
      numeroRighe=1;
      valoriDefault={'1','1'};
      datiInput = inputdlg(campi,titoloDLG,numeroRighe,valoriDefault);
      datiInput = char(datiInput);
      T=str2num(datiInput(1,:)); % GRANDEZZA INTERVALLO
      N=str2num(datiInput(2,:)); % NUMERO NODI

      % DATI DI ASCISSE E ORDINATE PER FUNZIONE E PER CAMPIONI
      x=linspace(-T/2,T/2,401); %DEFINISCE DOMINIO (ASCISSE fun)
```

```
y=feval(fun,x); % VALUTA LA FUNZIONE NEL DOMINIO (ORDINATE fun)
xj=T/N.*(-N/2:N/2)'; % ASCISSE DEI CAMPIONI
yj=feval(fun,xj); % CALCOLO VALORI DEI CAMPIONI

% PARTICOLARE MATRICE f DOVE LA PRIMA RIGA E' LA MEDIA DEI
% VALORI DEI CAMPIONI AGLI ESTREMI, MENTRE LA SECONDA RIGA
% E' COSTITUITA DAI VALORI FRA ESSI INTERMEDI
f=[.5*(yj(1)+yj(end));yj(2:end-1)];

% APPLICO SU TALE MATRICE L'ALGORITMO fft() E INVERTO CON
% fftshift() LE DUE META' DELL'ARRAY RISULTATO
numFT=fftshift(fft(f));

% COMPONENTI DELLA F.T. numFT. IL PRIMO DEVE ESSERE UGUALE
% ALL'ULTIMO, QUINDI LO AGGIUNGO IN CODA E SCALO TUTTI I
% GLI ELEMENTI DI UN FATTORE T/N
numFT=[numFT; numFT(1)]*T/N;

% CONTROLLO IF - COMPONENTI F.T. NEGATIVE E POSITIVE
% STABILISCE A PARTIRE DA QUALE COMPONENTE BISOGNA
% COMINCIARE A CAMBIARE IL SEGNO ALLE COPPIE ALTERNE.
% STABILISCO QUESTO CONTROLLANDO SE LA SEMIAMPIEZZA E' PARI O
% DISPARI
semiIntervallo=N/2;

if mod(semiIntervallo,2)==0 % SE IL SEMINTERVALLO E' PARI
      numFT(2:2:end) = -numFT(2:2:end); % PARTI DA 2° COMPONENTE
else % ALTRIMENTI
      numFT(1:2:end) = -numFT(1:2:end); % PARTI DALLA PRIMA
end

% DISEGNO IN NERO I PUNTI DELLA FUNZIONE fun DA APPROSSIMARE x,y;
% IN PALLINI ROSSI I CAMPIONI DI APPROSSIMAZIONE
% GLI ASTERISCHI VERDI SONO I PUNTI DELLA FUNZIONE f DA TRASFORMARE
figure
plot(x,y,xj,yj,'ro',xj(1:end-1),f,'gp')

% CREA STRINGA PER TITOLO GRAFICO
titolo=['Funzione f(x) = ' fString ' - e suoi Campioni' ];

% IMPOSTA STRINGA COME TITOLO
title(titolo);
```

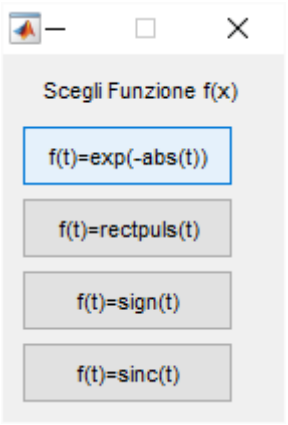
```
% CALCOLO FREQUENZE nu
nu=(-N/2:N/2)'/T; % FREQUENZE DI OGNI CAMPIONE
Omega=N/T; % OMEGA E' AMPIEZZA INTERVALLO PER OGNI CAMPIONE

% DISEGNO LA FUNZIONE SIMBOLICA DI F.T. ANALITICA (FREQ. CIRCOLARE)
figure
ezplot(abs(Fv), [-Omega/2 Omega/2]);
hold on

% DISEGNO FUNZIONE NUMERICA DI F.T. ATTRAVERSO LE FREQUENZE
plot(nu, abs(numFT),'or:');
axis tight

% CREA STRINGA PER TITOLO GRAFICO
titolo=['Trasformata di Fourier per f(x) = ' fString];

% IMPOSTA STRINGA COME TITOLO
title(titolo);
legend('F.T. Analitica','F.T. Numerica');

% DISEGNO LA FUNZIONE SIMBOLICA DI F.T. ANALITICA (FREQ. ANGOLARE)
figure

% RICALCOLIAMO CON OPPORTUNA SOSTITUZIONE, NUOVAMENTE LA F(w)
% PER CONFRONTARE RISULTATI IN FUNZIONE DI v E w
Fw=subs(Fv,v,w/(2*pi));
ezplot(abs(Fw), [-Omega/2 Omega/2]);
hold on

% DISEGNO FUNZIONE NUMERICA DI F.T. ATTRAVERSO LE FREQUENZE
plot(nu, abs(numFT),'or:');
axis tight

% CREA STRINGA PER TITOLO GRAFICO
titolo=['Trasformata di Fourier per f(x) = ' fString];

% IMPOSTA STRINGA COME TITOLO
title(titolo);
legend('F.T. Analitica','F.T. Numerica');
```

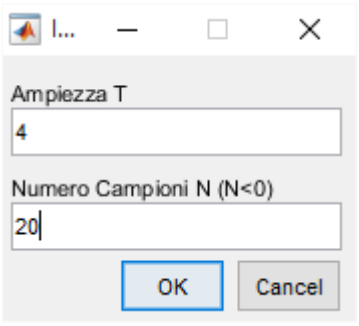
```
      % CRITERI DI SELEZIONE USCITA PER GESTIRE CICLO WHILE
      % ATTRAVERSO UNA QUESTION DIALOG BOX
      uscita=questdlg('Vuoi uscire?','Continua...');

end % END WHILE

end % END FUNCTION
```

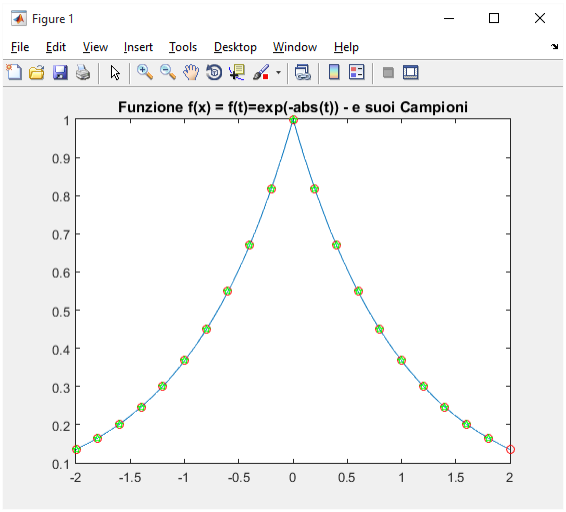
```
%   Matematica Applicata e Computazionale
%
%   Approssimazione Trasformaata di Fourier
%
%        Programma elaborato da
%
%           Giovanni DI CECCA
%               108/1569
%
%        http://www.dicecca.net
%
%                © 2016
%
%  GNU/GPL License
%
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function scelta_funzioneFT.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [f, Fv, fString] = scelta_funzioneFT(t,v,w)

% FUNZIONI TEST
cellFunzioni={'f(t)=exp(-abs(t))';
'f(t)=rectpuls(t)';
'f(t)=sign(t)'
'f(t)=sinc(t)'};

funzioneScelta = menu('Scegli Funzione f(x)',cellFunzioni);

switch funzioneScelta
      case 1
            % DECADENZA PARI f(x)=e^(-|t|)
            f=@(t) exp(-abs(t));
            ft=sym(f);
            applicaSubs=0;

      case 2
            % IMPULSO RETTANGOLARE f(t)=rectpuls(t)
            f=@rectpuls;
            ft=heaviside(t+0.5)-heaviside(t-0.5);
```

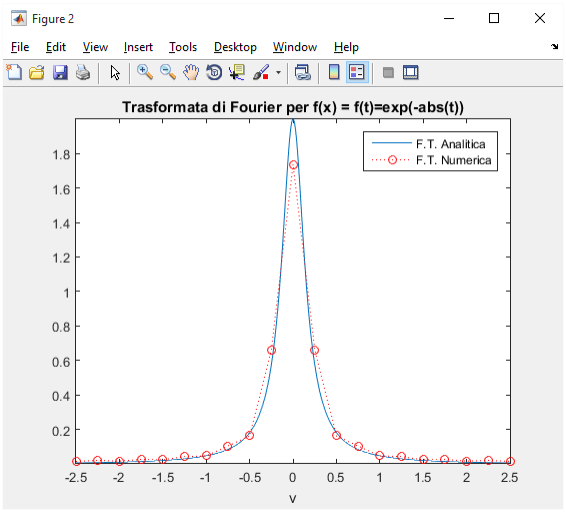
```
            applicaSubs=0;

      case 3
            % SEGNO f(t)=sign(t)
            f=@sign;
            ft=2*heaviside(t)-1;
            applicaSubs=0;

      case 4
            % RAPPORTO SENO E SUO ARGOMENTO
            f=@sinc; %sin(t)/t
            applicaSubs=1;
end

% SE LA FUNZIONE f NON E' sinc, DEFINISCI Fv CON ALGORITMO CLASSICO
if applicaSubs==0
      % TRASFORMATA DI FOURIER ANALITICA
      % CREA LA F.T ANALITICA DELLA FUNZIONE fun IN SYMBOLIC,
      % SOTTO FORMA ANGOLARE F(w), CHE COME SAPPIAMO GENERA UNA
      % APPROSSIMAZIONE SCORRETTA TROPPO LONTANA DALLA CURVA CHE SI VUOLE
      % RICOSTRUIRE
      FTsym=fourier(ft); %SI OTTIENE F(w) IN FREQ. ANGOLARE

      % ... PER SOSTITUZIONE, CALCOLA LA F.T. ANALITICA DI fun SOSTITUENDO
      % LA FREQUENZA ANGOLARE DELLA IN FORMA IN FREQ. CIRCOLARE F(v)
      % SOSTITUIENDO w=2*PI*v
      Fv=subs(FTsym,w,2*pi*v);
else % ALTRIMENTI...
      % ... DEFINISCI Fv PER f(x)=sinc
      Fv=heaviside(v+0.5)-heaviside(v-0.5);

end % END IF-ELSE

fString=cellFunzioni{funzioneScelta};

end % END FUNCTION
```

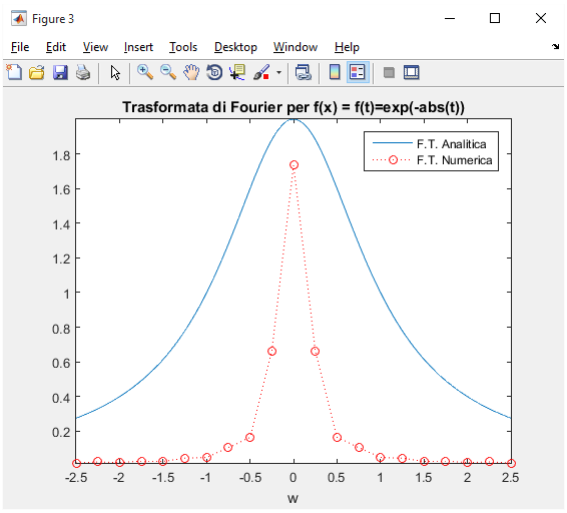
Esempio d'uso











Indice

LIBERTÀ EGUAGLIANZA

MONITORE NAPOLETANO

Fondato nel 1799 da
Carlo Lauberg ed Eleonora de Fonseca Pimentel

Rifondato nel 2010
Direttore: Giovanni Di Cecca

Anno CCXXII

Contatti

C.Ph.: +39 392 842 76 67

www.monitorenapoletano.it

info@monitorenapoletano.it