# Laboratorio Calcolo Numerico

## Esercizio 1

Si vuole determinare un polinomio interpolatore della funzione di Runge

$$f(x) = \frac{1}{1+x^2}$$

nell'intervallo $I = [-5, 5]$. Considerata la forma di Newton del polinomio interpolante, si costruisca una tabulazione della funzione data costituita da 11 nodi equidistanti nell'intervallo I, e si memorizzino i nodi in una vettore `xn` ed i corrispondenti valori della funzione nel vettore `fxn`.

Si scriva una funzione di nome `polnewton` che abbia la seguente intestazione e che calcoli i coefficienti della base di Newton con le differenze divise.

```
function c = polnewton (x,y)
% POLNEWTON Calcola i coefficienti del polinomio interpolatore
%           utilizzando la forma di Newton con le differenze divise
%
% Uso:
% c = polnewton (x,y)
%
% Dati di ingresso:
% x     vettore dei nodi
% y     vettore dei valori della funzione da interpolare nei nodi
%
% Dati di uscita:
% c     vettore colonna dei coefficienti ordinati per indici
%       crescenti (c_0, c_1, ... )
```

Si definisca poi una function di nome `hornerN` che, dati i coefficienti $c_i$ ed i nodi, permetta di valutare un polinomio interpolatore $P(x)$ in un valore $x^*$. La function avrà la seguente intestazione

```
function fxstar = hornerN (x,c,xstar)
% HORNERN Calcola il valore del polinomio interpolatore in x^*
% utilizzando la forma di Newton e l'algoritmo di Horner
%
% Uso:
% fxstar = hornerN (x,c,xstar)
%
% Dati di ingresso:
% x vettore dei nodi
% c vettore dei coefficienti della forma di Newton
% ordinati per indici crescenti (c_0, c_1, ... )
% xstar valore in cui si vuole valutare il polinomio
%
% Dati di uscita:
% fxstar valore di P(x^*)
```

Si scriva poi uno script che calcoli il valore del polinomio di interpolazione in 201 valori dell'intervallo $I$, per poterlo rappresentare graficamente (linea tratteggiata rossa).

Nello stesso grafico si rappresentino anche i punti della tabulazione (con un cerchietto rosso) e la funzione data (linea intera nera). Il risultato dovrà essere quello della seguente figura:

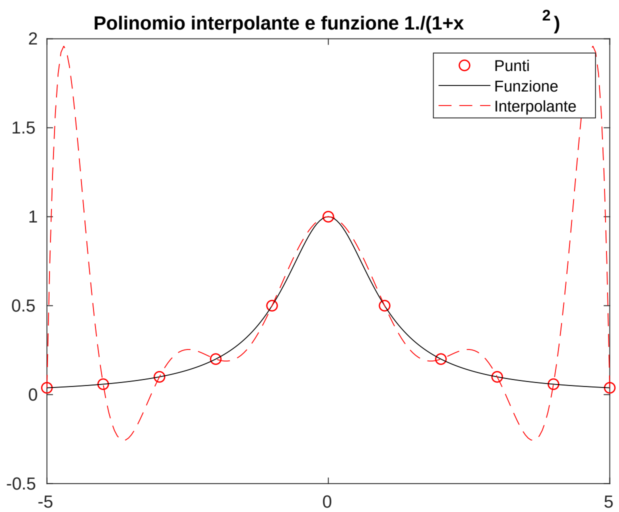


Figura 1: Grafico che illustra l'interpolazione della funzione di Runge in $[-5,5]$ su 11 nodi equispaziati.

## Traccia per risolvere l'esercizio 1

Per scrivere la funzione polnewton si deve prendere come base l'algoritmo di pag. 292. L'algoritmo va attentamente letto anche nella descrizione iniziale per capire il significato delle differenti variabili. Poi tale algoritmo va modificato per rispondere alle richieste del testo dell'esercizio (in uscita si vuole, ad esempio, solamente la prima riga dello schema e non tutta la tabella) ed in ingresso non viene dato il numero dei nodi (componenti dei vettori) ed anche per adattarsi alla sintassi Matlab che utilizza solo indici di riga e di colonna che iniziano da 1 (e non da zero).

Si noti che nell'algoritmo originale si usa, per lo schema triangolare, una matrice indicata con la variabile $C$ ed all'interno con la notazione $c_{i,j}$ si rappresenta l'elemento di riga $i$ e colonna $j$ di tale matrice. Della matrice, essendo uno schema triangolare, si usa solo una parte triangolare.

Per non confondere la matrice $C$ con il vettore $c$ richiesto nelle specifiche della function polnewton, chiamiamo nell'algoritmo tale matrice `ctab`. Inoltre la variabile locale interna `n1` contiene il numero di nodi $n+1$. Ovviamente anche i valori assunti dalle variabili nei cicli ad indice fisso dovranno essere modificati.

A questo punto, gli studenti, dovrebbero scrivere prima l'algoritmo modificato come segue, e poi tradurlo in una function Matlab di nome

**[c]= polnewton(x, y)**

$n1$ = lunghezza vettore $x$
**if** $n1 \neq$ lunghezza vettore $y$
  Errore. La tabulazione non contiene lo stesso numero di componenti. Stop.
**end if**
**for** $i = 1, \ldots, n1$
  $ctab(i,1) = y(i)$
**end for** i
**for** $j = 2, \ldots, n1$

**for** $i = 1, \ldots, n1 - j + 1$

$ctab(i,j) = (ctab(i+1, j-1) - ctab(i, j-1))/(x(i+j-1) - x(i))$

**end for** i

**end for** j

c = vettore colonna contenente la prima riga di ctab

Per tradurre in Matlab la prima istruzione si veda il comando Matlab length e per l'ultima, si usi il comando `c = ctab(1,:)` che estrae la riga 1 di una matrice.

Vediamo ora come impostare il secondo algoritmo per poterlo tradurre nella function `hornerN`. L'algoritmo si trova a pag. 271 del libro di testo. Come in precedenza si deve prima modificare il seguente algoritmo originale per adattarlo alle esigenze del Matlab per quanto riguarda gli indici delle componenti dei vettori.

Algoritmo originale:

**[u]= hornerN** $(x, c, x^*)$

$u = c_n$

**for** $j = n-1 \ldots 0, \mathbf{step} - 1$

$u = u(x^* - x_j) + c_j$

**end for** $j$

dove

$$\begin{aligned} d_j &= (x^* - x_j) \\ u &= P_n(x^*) = c_0 + c_1 d_0 + c_2 d_0 d_1 + \ldots + c_n d_0 d_1 \ldots d_{n-1} = \\ &= (\ldots(((c_n)d_{n-1} + c_{n-1})d_{n-1} + c_{n-2})d_{n-2} + \ldots + c_1)d_0 + c_0. \end{aligned}$$

Ecco l'algoritmo modificato:

**[u]= hornerN (x, c, xstar)**

$n1$ = lunghezza del vettore x

$n1 \neq$ lunghezza del vettore **c**

Errore. Inconsistente numero di componenti. Stop.

$u = c(n1)$

**for** $j = n1 - 1 \ldots 1, \mathbf{step} - 1$

$u = u * (xstar - x(j)) + c(j)$

**end for** $j$

A questo punto basta tradurlo in Matlab, rispettando i nomi indicati per i parametri di ingresso e di uscita.

Vediamo uno schema per lo script (ovviamente mancano le istruzioni Matlab che dovranno essere inserite dagli studenti!

`Script interp`

```
%=======================================================================
% Script per interpolazione con la forma di Newton
% Necessita delle function polnewton e hornerN
%=======================================================================
% Definisce la funzione
% ..................................
% Definisce l'intervallo [a,b]
% ..................................
% Definisce il numero di nodi
```

```
% ................................
% Definisce la tabulazione dei punti (ascisse equispaziate ed ordinate
% come valori della funzione nei nodi definiti)
% ................................
% Calcola i coefficienti del polinomio con la base di Newton usando
%   la function polnewton
% ................................
% Grafici
% Definisce le ascisse per i grafici (201 punti equispaziati)
% ................................
% Valuta la funzione nelle 201 ascisse
% ................................
% Valuta il polinomio nelle 201 ascisse (ciclo for che usa
%     la function hornerN)
% ................................
% Creazione della figura che contiene i punti della tabulazione
%   la rappresentazione della funzione originale
%   la rappresentazione del polinomio interpolante
% ................................
% Creazione della figura che contiene la funzione errore
% ................................
```

### Esercizio 2

Si vari (aumentando e diminuendo) il numero di punti di interpolazione (e quindi il grado del polinomio interpolante) e si cerchi di capire cosa accade.

### Esercizio 3

Si produca anche una figura che rappresenti nello stesso intervallo l'asse x e la funzione errore definita come

$$E(x) = f(x) - P(x).$$

ottenendo la seguente figura

### Esercizio 4

Si calcoli l'errore commesso nell'interpolare la funzione di Runge $f(x) = \dfrac{1}{1+x^2}$ in $[-5, 5]$ su nodi equispaziati con polinomi di grado $n$, con $n = 1, 3, 5, \ldots, 31$. Si risolva l'esercizio modificando lo script precedentemente creato aggiungendo un ciclo for che faccia variare il valore di $n$. Per ciascun grado $n$ del polinomio si calcolino i nodi e i valori della funzione $f(x)$ in essi, si interpoli la funzione in tali punti e si valuti la norma infinito dell'errore commesso (si consulti la help dell comando `norm` di Matlab). Si produca infine una tabella su file contenente il grado del polinomio e la norma infinito degli errori ottenuti per ogni polinomio considerato. Si chiami il file dei risultati `risequis.txt`.
Per scrivere su file in Matlab basta aprire il file per scrittura con il comando `fopen` e passare come primo parametro al comando `fprintf` il numero di unità associato al file. Segue un esempio:

```
fid=fopen('risequis.txt','w');
...
fprintf(fid, ...);
...
fclose(fid);
```

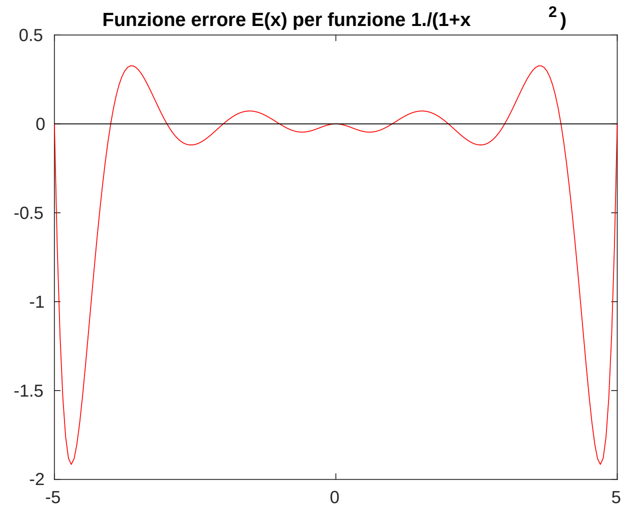


Figura 2: Errore commesso interpolando la funzione di Runge in $[-5, 5]$ con il polinomio di interpolazione ottenuto su 11 nodi equispaziati.

Osservando l'andamento degli errori al crescere del grado del polinomio che cosa si può dire sulla convergenza dell'interpolante calcolato su nodi equispaziati alla funzione $f(x)$?

## Esercizio 5

Si scriva la function `chebgauss.m` che calcola il vettore dei nodi di Chebyshev in un intervallo $[a, b]$ utilizzando la formula seguente:

$$x_i = \frac{(a+b)}{2} + \frac{(b-a)}{2} t_i, \; i = 0, \dots, n \tag{1}$$

con

$$t_i = \cos\left(\frac{2i+1}{2n+2}\pi\right), \; i = 0, \dots, n; \tag{2}$$

La sintassi della function dovrà essere

```
function xc = chebgauss(a,b,m)
```

## Esercizio 6

Si ripeta l'esercizio 4 utilizzando i nodi di Chebyshev al posto dei nodi equspaziati. Questa volta si chiami il file contenente la tabella dei risultati `risCheb.txt`.

Osservando l'andamento degli errori al crescere del grado del polinomio che cosa si può dire sulla convergenza dell'interpolante calcolato sui nodi di Chebyshev alla funzione $f(x)$?

## Esercizio 7

Risultati analoghi a quelli degli esercizi precedenti si possono ottenere calcolando e valutando il polinomio di interpolazione, rispettivamente, con le due funzioni Matlab seguenti:

```
POLYFIT Fit polynomial to data.
     P = POLYFIT(X,Y,N) finds the coefficients of a polynomial
```

```
    P(X) of degree N that fits the data Y best in a least-squares sense.
    P is a row vector of length N+1 containing the polynomial coefficients
      in descending powers,
    P(1)*X^N + P(2)*X^(N-1) +...+ P(N)*X + P(N+1).
```

che permette di determinare un vettore che contiene i coefficienti del polinomio interpolante rispetto alla base canonica;

```
POLYVAL Evaluate polynomial.
      Y = POLYVAL(P,X) returns the value of a polynomial P evaluated at X.
      P is a vector of length N+1 whose elements are the coefficients of the polynomial
        in descending powers.
      Y = P(1)*X^N + P(2)*X^(N-1) + ...  + P(N)*X + P(N+1)
```

che valuta un polinomio di cui si conoscono i coefficienti rispetto alla base canonica su tutti i valori precedentemente memorizzati su di un vettore. Si producano le stesse figure precedentemente richieste (che risulteranno praticamente uguali) e poi si cerchi di strutturarle con il comando `subplot` per ottenere la figura mostrata in 3.

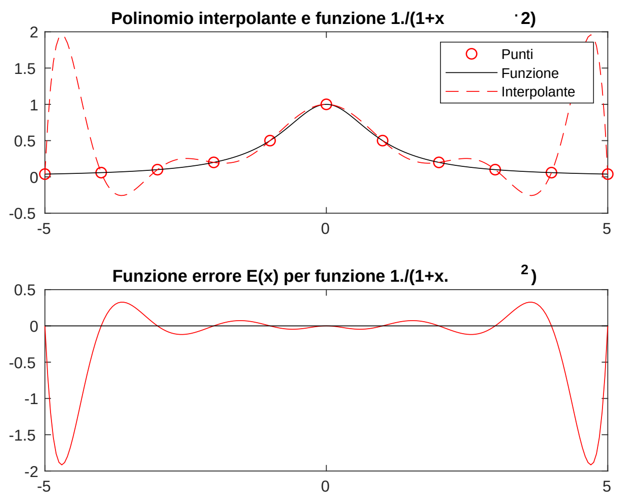


Figura 3: Interpolazione della funzione di Runge in $[-5, 5]$ su 11 nodi equispaziati (top) ed errore corrispondente (bottom).