UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
INDUSTRIALE

DIPARTIMENTO
MATEMATICA
DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

MATLAB STEP BY STEP

Materiale realizzato da Michela Redivo Zaglia

con il contributo di E. Bachini, L. Bruni, W. Erb, A. Larese, F. Piazzon

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE

DIPARTIMENTO MATEMATICA

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

Laboratorio di Calcolo Numerico LAB 6

Interpolazione polinomiale

Docenti: E. Bachini, L. Bruni

Email: elena.bachini@unipd.it Email: bruni@math.unipd.it

10 aprile 2024

Outline

Comandi utili

error e warning

- `error(stringa)` causa l'immediata interruzione dell'esecuzione di un programma e l'output a video del messaggio di errore (rosso) contenuto nella stringa
- `warning(stringa)` causa l'output video di un messaggio di warning (giallo) che informa l'utente delle possibili problematiche nell'esecuzione di un programma (es: possibile instabilità dell'algoritmo con i dati caricati).

break

- `break` causa l'immediata uscita da un ciclo `for`/`while` *senza terminare l'iterazione in corso*. Tipicamente utilizzato all'interno di una struttura `if`. Da usare solo se non vi sono alternative semplici per ottenere gli stessi risultati.

Esempio: questi due programmi sono quasi uguali. Perché?

```
m=13;
imax=5;
i=0;
while m>0
    m=m-2;
    i=i+1;
    if i>=imax
        break
    end
    disp(m)
end
```

```
m=13;
imax=5;
i=0;
while (m>0) && (i<imax)  %% oppure and(m>0, i<imax)
    m=m-2;
    disp(m)
    i=i+1;
end
```

return

- `return` in uno script (o in una function) eseguito o chiamato da command window causa l'interruzione dell'esecuzione del programma in corso e il ritorno al command prompt.
- `return` in uno script (o in una function) eseguito o chiamato da un programma chiamante (script o function) causa l'interruzione dell'esecuzione del programma in corso e il ritorno al programma chiamante.

N.B.: In una function a seguito di un comando `return` potrebbero non essere stati assegnati tutti gli output. Questo *causa errore* solo qualora la function venga chiamata richiedendo di assegnare ad una variabile l'output che non è stato calcolato.

return - Esempio

Esempio:

```
% FUNCTION di prova return

function [flag,sq] = provareturn(x)
flag = 0;
% viene calcolato sqrt solo se in valore e' positivo
if x>0
    sq = sqrt(x);
    flag = 1;
else
    return
end
```

```
>> flag = provareturn(-1)
flag =
     0

>> [flag,sq] = provareturn(-1)
Output argument "sq" (and maybe others) not assigned during call to "
    provareturn".
```

Interpolazione Polinomiale con Matlab

Interpolazione polinomiale

Problema

Dati $x_i \in [a, b]$, $i = 0, 1, \ldots, n$ con $x_i \neq x_j \ \forall i \neq j$ e $y_i = f(x_i) \in \mathbb{R}$, $i = 0, 1, \ldots, n$ trovare p polinomio di grado al più n che *interpoli* i dati (x_i, y_i), $i = 0, 1, \ldots, n$:

$$p(x_i) = y_i, \ i = 0, 1, \ldots, n.$$

Risultato fondamentale

Se il numero di punti $(n + 1)$ è uguale al grado del polinomio (n) più 1, il polinomio interpolante *esiste ed è unico*.

Come possiamo calcolare p con Matlab?

Calcolo dei coefficienti di p

Calcolare un polinomio vuol dire calcolare la sua rappresentazione su una base, tipicamente quella canonica $\{1, x, x^2, \ldots, x^n\}$, per poterlo scrivere come

$$p(x) = \sum_{k=1}^{n+1} c_k x^{n-k+1} = c_1 x^n + c_2 x^{n-1} + \cdots + c_{n+1}$$

Attenzione agli indici!!

Per farlo, occorre risolvere il sistema di Vandermonde

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \begin{pmatrix} c_{n+1} \\ c_n \\ \vdots \\ \vdots \\ c_1 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ \vdots \\ y_n \end{pmatrix}$$

Questa fatica la lasciamo alla function matlab `polyfit`.

Chiamata di `polyfit`

```
c = polyfit(x,y,n)
```

Input

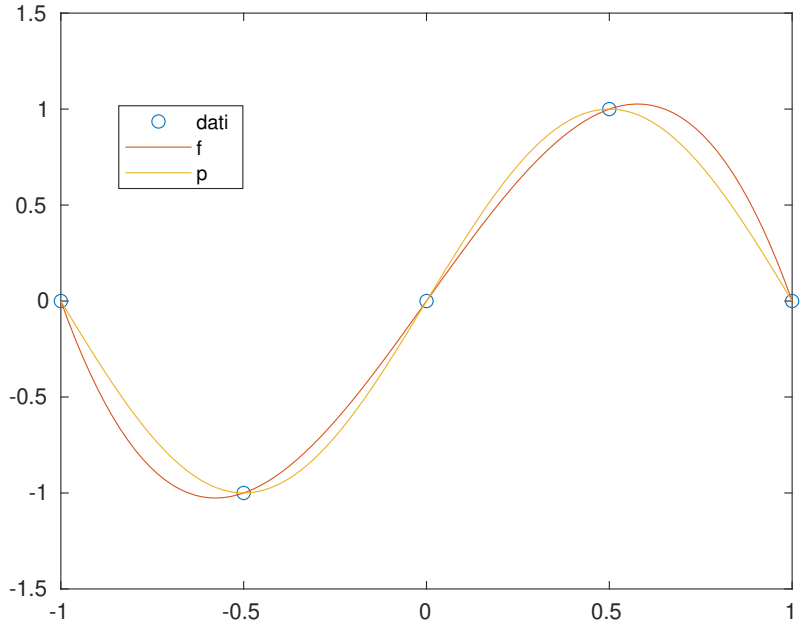
- `x` nodi di interpolazione. Vettore double `[n+1,1]` o `[1,n+1]` di punti distinti
- `y` valori da interpolare. Vettore double `[n+1,1]` o `[1,n+1]`
- `n` grado polinomiale. Intero positivo

Output

- `c` coefficienti del polinomio interpolatore, *ordinati in modo decrescente rispetto al grado del monomio*. Vettore riga `[1,n+1]` di double

Esempio: interpolazione di campionamento di funzione

```
n = 4;
f = @(x) sin(pi*x);
x = linspace(-1,1,n+1);     % discretizzazione di [-1,1]
y = f(x);
c = polyfit(x,y,n);
xplot = linspace(-1,1);
p = (xplot'.^(n:-1:0))*c'; % poco efficiente (useremo polyval)
plot(x,y,'o'); hold on
plot(xplot,p);
plot(xplot,f(xplot));
legend('dati','f','p')
```



Osservazioni e raccomandazioni importanti

- Attenzione: Per calcolare un interpolante è necessario che il numero di nodi e di valutazioni coincidano e che i nodi `x` siano a 2 a 2 distinti. Si noti che se tali condizioni non sono soddisfatte, il Matlab **non segnala un errore**!
- Non confondere i *nodi di interpolazione* `x` (che devono essere distinti e `n+1`) con i *punti di valutazione* `xplot` che abbiamo usato per rappresentare graficamente il polinomio interpolante!
- Abbiamo plottato i dati solo con marker perchè la spezzata che li congiunge (che si sarebbe ottenuta con `plot(x,y)`) sarebbe stata priva di significato.
- Ricordare l'ordine dei coefficienti forniti in output da `polyfit`.

Valutare un polinomio con `polyval`

Usando la chiamata polyval possiamo evitare di ricordare l'ordine dei coefficienti (Matlab lo fa per noi). Ma come si usa polyval?
Chiamando `y = polyval(c,x)`. In Matlab è implementato in modo efficiente.

Input

- `x` vettore (riga o colonna) di lunghezza arbitraria contenente i punti di valutazione
- `c` vettore (riga o colonna) dei coefficienti di un polinomio, ordinati come in `polyfit`

Output

- `y` vettore riga o colonna con `size(y)=size(x)` di valutazioni del polinomio sui punti `x`.

Osservazione importante

Nella chiamata `polyval(c,x)` non viene passato in input il grado. Questo è implicitamemte definito dalla lunghezza di `x`.
Contrariamente, nella chiamata `c=polyfit(x,y,n)` dobbiamo sempre passare il grado `n`. Perché?

`polyfit` è strumento per il fitting

Lo scopo principale di `polyfit` è quello di fare "fitting" di dati (ad esempio sperimentali), la possibilità di *interpolare* è solo una possibile applicazione che si ottiene solo quando `length(x)=n+1`. Per questo motivo la variabile `n` in input è richiesta.

Esempio

```
n = 4;
f = @(x) sin(pi*x);
x = linspace(-1,1,n+1);
y = f(x);
c = polyfit(x,y,n);
xplot = linspace(-1,1);
p = polyval(c,xplot); % usa schema di Horner
plot(x,y,'o'); hold on
plot(xplot,p);
plot(xplot,f(xplot));
legend('dati','f','p')
```

`polyval` può essere usato anche all'interno di una anonymous function

```
n = 4;
f = @(x) sin(pi*x);
xinterp = linspace(-1,1,n+1);
yinterp = f(xinterp);
c = polyfit(xinterp,yinterp,n);
xplot = linspace(-1,1);
p = @(x)polyval(c,x); % polyval e' usata per definire p
% (anonymous function)
pplot = p(xplot);
plot(x,y,'o'); hold on
plot(xplot,pplot);
plot(xplot,f(xplot));
legend('dati','f','p')
```

Esempio

Cerchiamo di interpolare la funzione $f(x) = e^x \sin(x)$ sull'intervallo $[-1, 1]$ su 10 punti equispaziati (quindi con un polinomio di grado 9).

- costruiamo il vettore di 10 punti equispaziati con $x = linspace(-1, 1, 10)$;
- definiamo il function handle $f = @(t)exp(t).*sin(t)$;
- valutiamo la funzione f nei 10 punti x: $F = f(x)$;
- calcoliamo i coefficienti del polinomio interpolatore $p(x)$ con $c = polyfit(x, F(x), 9)$;
- valutiamo il polinomio $p(x)$ in un vettore X con tanti punti (tipo $X = linspace(-1, 1, 1000)$) con $P = polyval(c, X)$;
- possiamo plottare il grafico con $plot(X, P)$.

Confronto tra diversi metodi

Vogliamo confrontare il risultato del problema di interpolazione risolto scegliendo la nostra funzione in diverse classi:

- interpolazione polinomiale, con un polinomio $p_n(x)$ di grado n che passa per i dati
- interpolazione lineare a tratti (di grado 1): i dati vengono interpolati da una retta spezzata
- interpolazione *spline*: rientra nelle interpolazioni a tratti, in questo caso il risultato è un polinomio di grado 3 che passa per i dati

Interpolazione polinomiale globale: `polyfit/polyval`

Avremo bisogno dei seguenti comandi Matlab:

- `polyfit`: prende in input i dati e il grado del polinomio desiderato, e restituisce i coefficienti del polinomio interpolatore
- `polyval`: prende in input i coefficienti di un polinomio (attenzione all'ordine degli indici!) e un punto di valutazione, e valuta il polinomio nel punto. Utile: può essere usato per valutare direttamente *in più punti* (usando opportunamente i vettori) – ed è conveniente farlo!

Interpolazione a tratti: `interp1` e `spline`

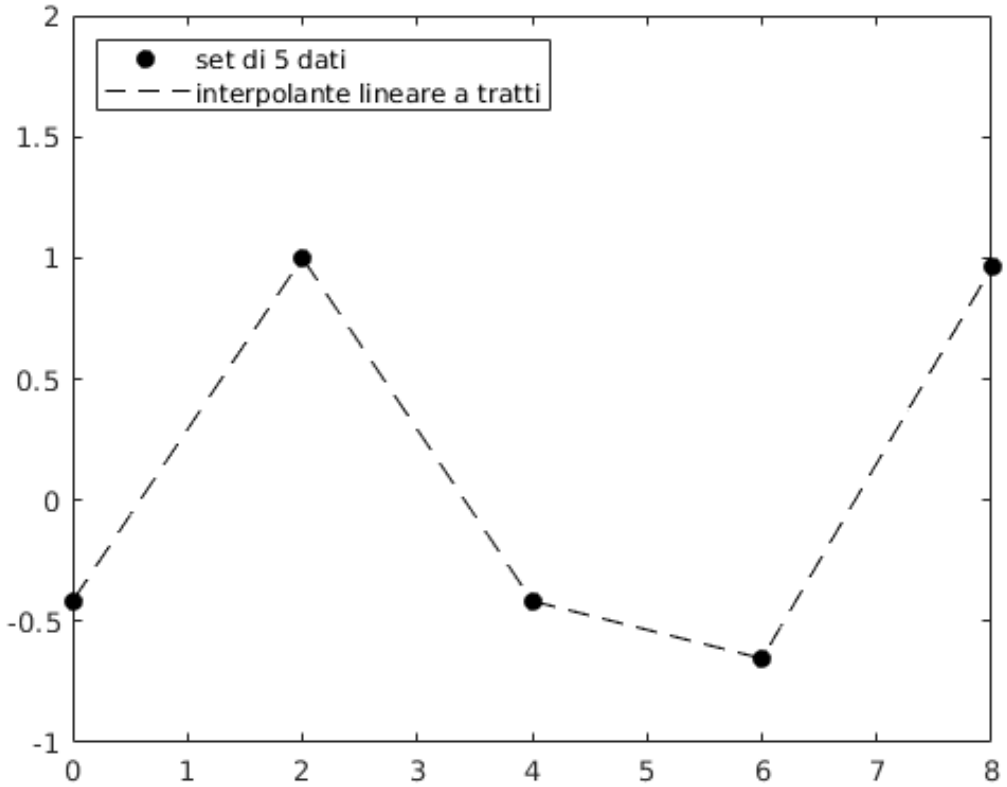
- `interp1`: prende in input i dati e i nodi di valutazione e restituisce il polinomio lineare a tratti (cioè di grado 1) che passa per i dati e *già valutato* nel nodo di valutazione.
- `spline`: prende in input i dati e i nodi di valutazione e restituisce le valutazioni del polinomio a tratti di grado 3 che passa per i dati.

NOTA: anche in questi casi, si può valutare direttamente *in più punti* (usando i vettori)!

Interpolazione polinomiale a tratti ('piecewise')

Dato un set di $n+1$ dati (coppie ordinate (x_i, y_i))

x_{data}	y_{data}
x_0	y_0
x_1	y_1
...	...
x_n	y_n

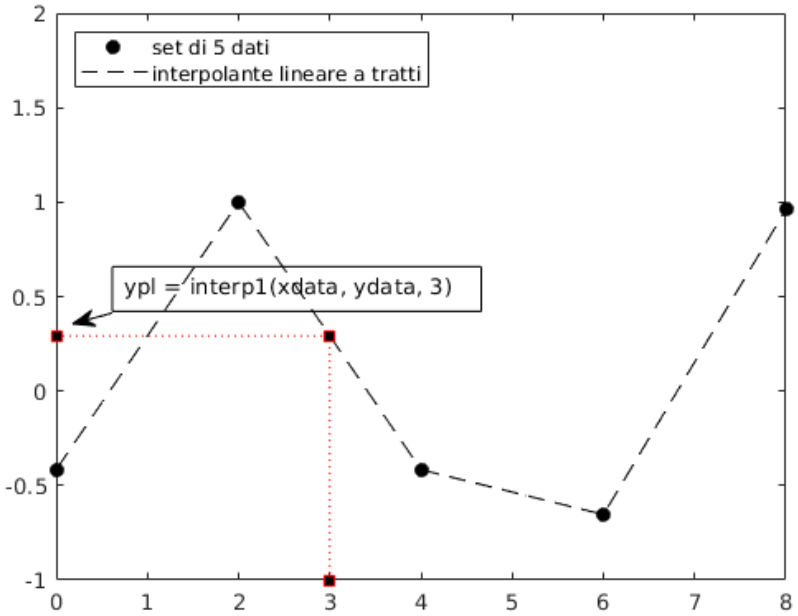


Per ogni coppia di punti (x_i, y_i) e (x_{i+1}, y_{i+1}) voglio trovare il polinomio $p^j(x)$ di grado m

Interpolazione lineare a tratti - interp1

Se voglio usare l'interpolazione a tratti posso usare il comando interp1

```
ypl = interp1(xdata, ydata, x);
```



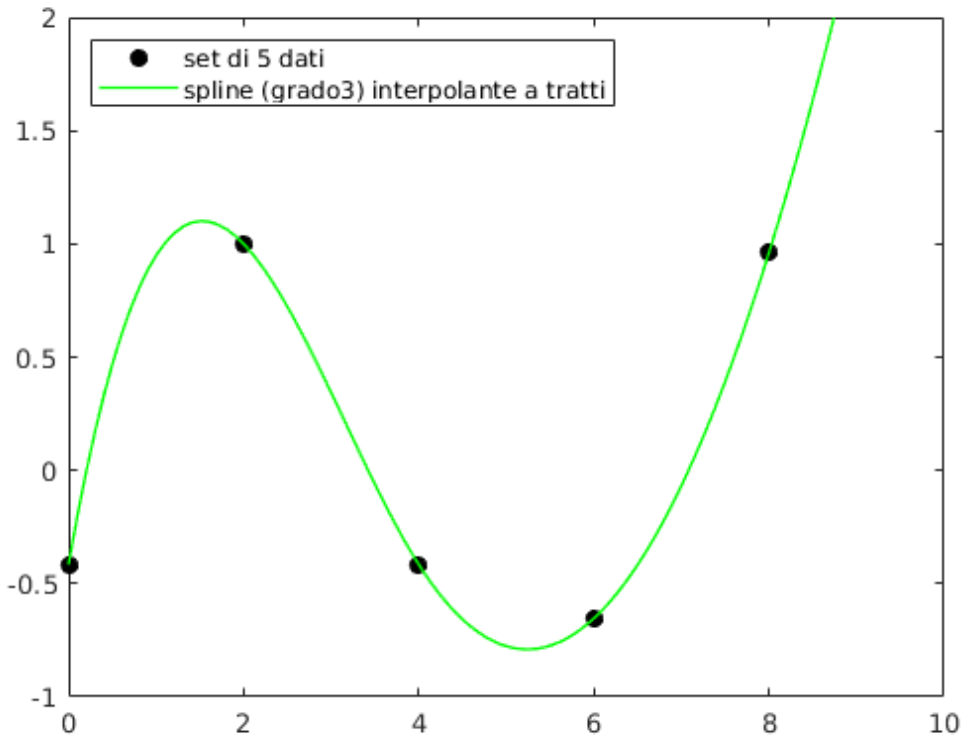
Nota.

Così facendo stiamo solamente richiedendo che la funzione interpolante sia continua, le sue derivate saranno discontinue.

Interpolazione polinomiale a tratti - spline

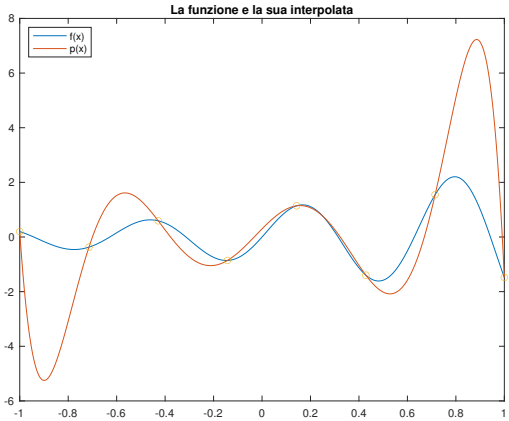
Se vogliamo richiedere anche continuità nelle derivate tra un intervallo e il seguente, possiamo usare le Splines.
Matlab ha la funzione `spline` che utilizza di default polinomi di terzo grado continui e che abbiano derivata continua tra un intervallo e l'altro.

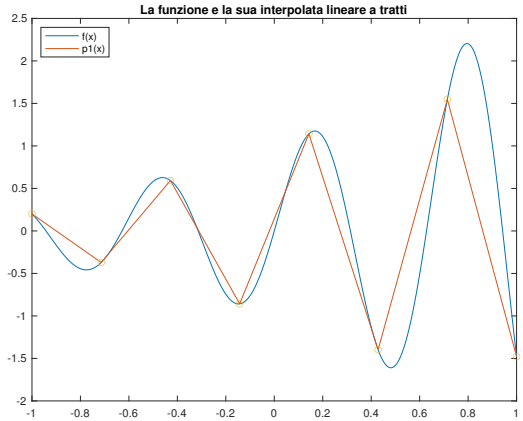
```
yspline = spline(xdata, ydata, x);
```

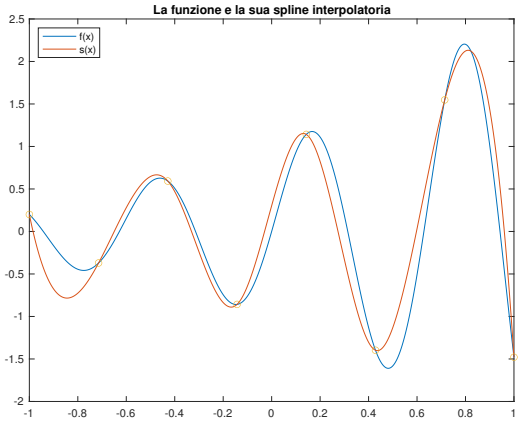


Confronto fra metodi

Consideriamo la funzione $f(x) = e^x \sin(10x)$ e il grado $n = 7$.

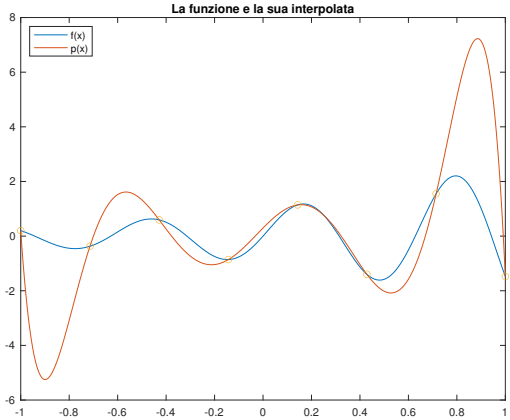


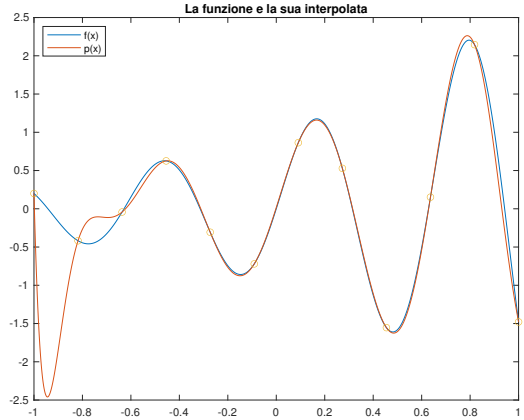


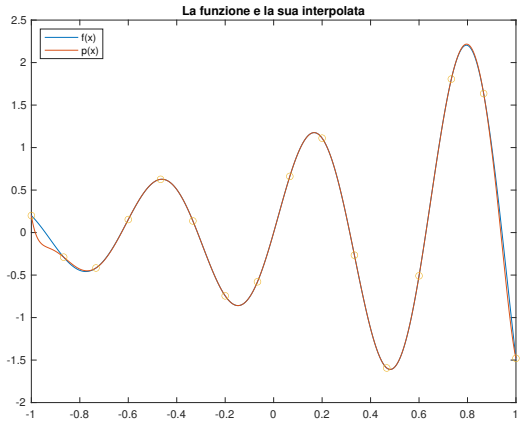


Polinomio interpolatore, interpolata lineare a tratti e spline interpolatoria.

Intuitivamente, più dati aggiungiamo (cioè più punti di valutazione mettiamo in gioco) più l'interpolata si avvicinia alla funzione che genera i dati.



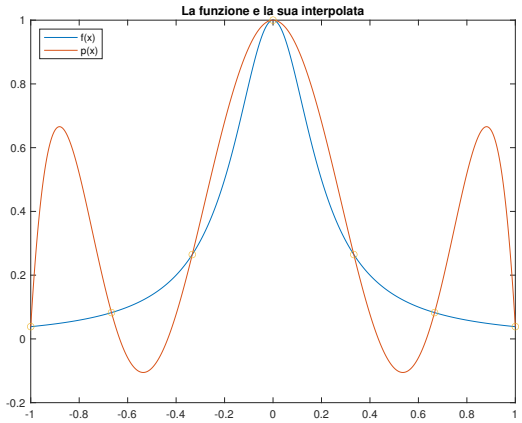


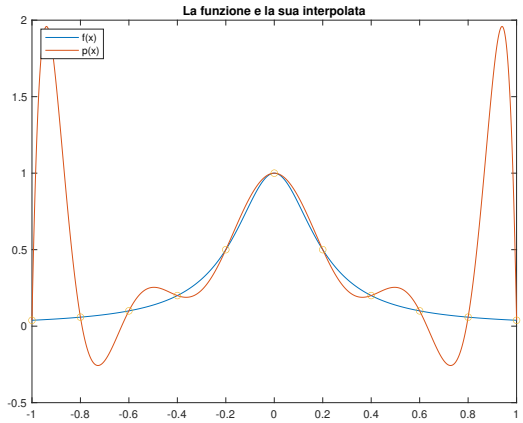


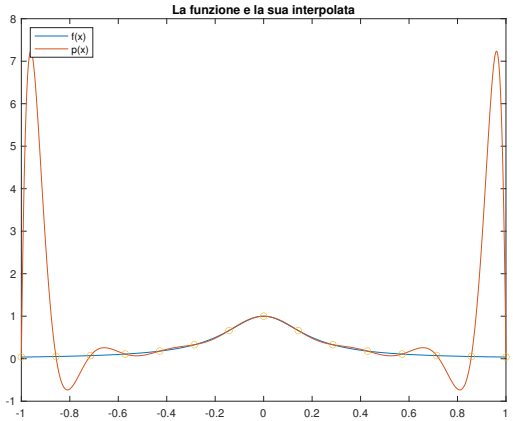
Qui abbiamo rappresentato le interpolate per gradi 7, 11 e 15.

Fenomeno di Runge

Ma questo sarebbe troppo bello per essere vero. La funzione di Runge $f(x) = \frac{1}{1+x^2}$, definita sull'intervallo $[-5, 5]$, presenta un'interpolata che peggiora se si aumenta il numero di punti equispaziati.



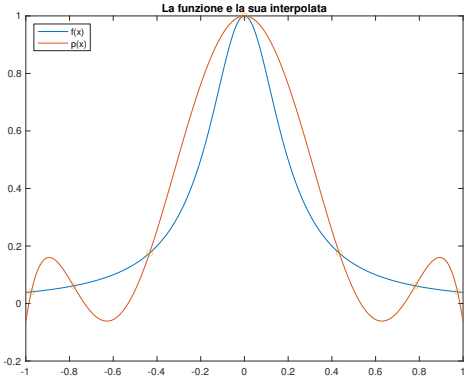


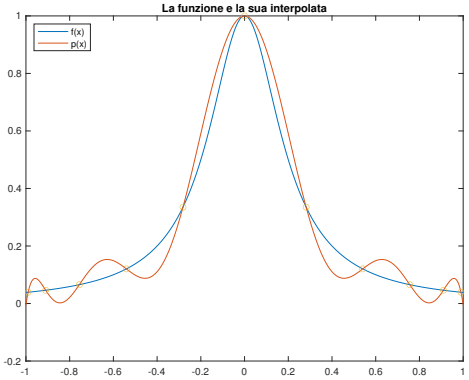


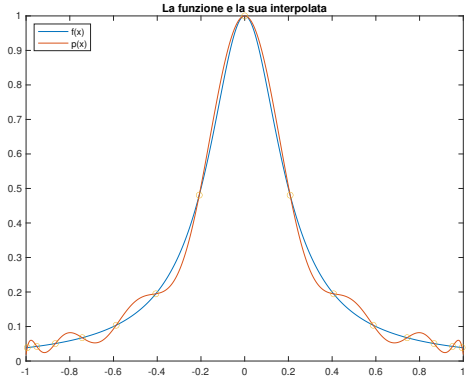
I punti di Chebyshev

Quindi cosa si fa, se vogliamo lavorare con l'interpolazione? Si cambiano i punti! Invece che considerare i punti equispaziati (cioè il linspace di Matlab), si considerano i punti di *Chebyshev* su $[-5, 5]$

$$x_i = -5\cos\left(\frac{2i+1}{2(n+1)}\pi\right), \quad i = 0, 1, \ldots, n.$$







Esercizio 1 (con consegna)

Si modifichi il template Matlab in modo tale che produca il grafico della funzione di Runge $f(x) = 1/(1+25x^2)$ su $[-1,1]$ al variare del numero di punti (dunque del grado), da $n = 2$ a $n = 20$. In dettaglio:

- Si considerino nodi equispaziati. Si stampino i grafici dell'interpolata di grado n, del polinomio interpolatore, dell'interpolata lineare a tratti e della spline cubica interpolante.
- Si ripeta il punto precedente coi nodi di Chebyshev $x_i = -\cos\left(\frac{2i+1}{2(n+1)}\pi\right)$, con $i = 0, 1, \ldots, n$.

Definiamo l'errore come il massimo della differenza tra il polinomio interpolatore e la funzione calcolati neii nodi di valutazione. In formule:

$$err = \texttt{max(abs(f(v)-p(v)))},$$

dove v è il vettore dei nodi di valutazione.

- Si costruisca un ciclo dipendente dal grado e in cui si salva l'errore per ogni grado. Si stampi il grafico.

Esercizio 1 - consegna

Si consegni una immagine in formato .jpeg che contiene:

- il grafico della funzione di Runge,
- il grafico del suo polinomio interpolatore di grado 15 rispetto ai nodi equispaziati,
- il grafico del suo polinomio interpolatore di grado 15 rispetto ai nodi di Chebyshev.

Esercizio 2 - `polyfit` e `polyval`

Date le seguenti funzioni:

1. $f(x) = e^{-x}$ con $x \in [0, 5]$,
   vettori (punti di interpolazione): $\hat{x} = \mathit{linspace}(0, 5, n+1), \quad \hat{y} = f(\hat{x})$
2. $f(x) = e^{-x}(1 + 0.3\sin(2x^2))$ con $x \in [0, 5]$,
   vettori: $\hat{x} = \mathit{linspace}(0, 5, n+1), \quad \hat{y} = f(\hat{x})$

Richiesta:

- Risolvere il problema di interpolazione sui punti $(\hat{x}, \hat{y})$, considerando i valori di $n = 5, 10, 20$
- Visualizzare in un unico grafico la funzione esatta $f(x)$ e i polinomi interpolanti ottenuti per i valori di n considerati.
  (CONSIGLIO: Utilizzare un vettore di punti di valutazione con m punti, ad esempio $m = 100$)