

Chebfun

Alvise Sommariva

Università degli Studi di Padova
Dipartimento di Matematica

10 marzo 2025

In questa nota, descriveremo i comandi di base dell'ambiente Chebfun, di rilevante importanza per l'analisi numerica ed in particolare per la teoria dell'approssimazione.

Utilizzeremo tale ambiente per studiare in particolare alcuni problemi quali:

- interpolazione in nodi equispaziati;
- interpolazione in nodi di Chebyshev;
- migliore approssimazione con algoritmo di Remez.

Le persone che non hanno installato Matlab sul proprio computer, possono farlo come segue:

- registrarsi all'account di Mathworks
<https://www.mathworks.com/mwaccount/register>,
- andare al link <https://www.mathworks.com/academia/tah-portal/universita-degli-studi-di-padova-31194939.html>,
- andare al frame **Desktop. Online. Mobile. Free through your school's license.** e cliccare **sign to get started**,
- inserire il proprio nome utente e password dell'universita' (SSO);
- si accettano o rifiutano delle regole;
- si clicca il bottone nel frame **Sign in to your existing MathWorks Account**;
- si inseriscono il proprio account/password di Mathworks;
- si scarica l'installer Matlab;
- si lancia l'installer Matlab seguendo le indicazioni.

Per una guida su Chebfun si consulti

<http://www.chebfun.org/docs/guide/>

mentre per installare l'ambiente Chebfun in Matlab sul proprio computer, si suggerisce di digitare su shell Matlab

```
unzip('https://github.com/chebfun/chebfun/archive/master.zip')  
movefile('chebfun-master', 'chebfun'),  
addpath(fullfile(cd, 'chebfun')), savepath
```

Se si copia e incolla tale codice, modificare gli accenti ' in verticali e dritti, come da tastiera. Alternativamente si copi e incolli direttamente il codice dalla homepage

<http://www.chebfun.org/download/>

senza modifiche.

Quanto faremo di seguito, è tratto ed adattato da

<http://www.chebfun.org/docs/guide/guide01.html>

L'idea di Chebfun consiste nell'idea che per rappresentare funzioni regolari spesso basta interpolare in 20 o 30 nodi di Chebyshev, ma che tale processo, qualora implementato adeguatamente, rimane stabile anche con migliaia o milioni di nodi di Chebyshev.

Avendo questo in mente, nella prima versione di Chebfun, ad una funzione f , l'ambiente `chebfun`, tramite procedure adattative, determinava il numero di punti $n + 1$ di Chebyshev per cui l'interpolante p_n era tale da approssimare f con un errore relativo di 10^{-15} .

Nelle versioni successive, l'ambiente è stato adeguato per poter trattare adeguatamente funzioni regolari a pezzi, determinando automaticamente i punti in cui suddividere l'approssimazione. Per capire questo punto, consideriamo la funzione

$$f(x) = |x - 0.3|$$

Digitiamo

```
f = chebfun('abs(x-.3)');
```

e otteniamo quale risposta

```
Warning: Function not resolved using 65537 pts. Have you tried 'splitting on'?
```

col significato che troppi punti vengono utilizzati per studiare la funzione f .

Se invece chiediamo all'ambiente `chebfun` di suddividere il dominio, il risultato migliora. Infatti

```
f = chebfun('abs(x-.3)', 'splitting', 'on');
```

non da' problemi. Per capire cosa sia successo e quali punti abbia usato `chebfun`, digitiamo

```
>> f
f =

    chebfun column (2 smooth pieces)
      interval      length      endpoint values
[   -1,    0.3]      2         1.3      0
[   0.3,    1]      2         0       0.7
vertical scale = 1.3      Total length = 4
>>
```

Evidentemente `chebfun` ha trovato il punto *critico* 0.3, e utilizzato l'interpolazione su due punti di Chebyshev scalati negli intervalli per determinare un'approssimazione della funzione f .

Esempio 1. Interpolazione in nodi equispaziati.

Consideriamo la funzione di Runge

$$f(x) = \frac{1}{1 + 25x^2}, \quad x \in [-1, 1].$$

e digitiamo

```
>> f=@(x) 1./(1+25*x.^2);  
>> F=chebfun(f)  
F =  
    chebfun column (1 smooth piece)  
      interval      length      endpoint values  
[      -1,       1]      189      0.038      0.038  
vertical scale = 1  
>>
```

La funzione di Runge è stata approssimata con un errore relativo di 10^{-15} utilizzando 189 nodi di Chebyshev.

Esempio 1. Interpolazione in nodi equispaziati.

Digitiamo nel file `esempio1.m`

```
function [eequi,echeb]=esempio1
d = [-1, 1]; % INTERVALLO.
ff = @(x) 1./(1+25*x.^2); % FZ. RUNGE
f=chebfun(ff,d);
eequi=[];
echeb=[];
nn=1:1:100;
for n=nn
    x = linspace(d(1), d(2), n+1); % (n+1) NODI EQUISPAZIATI.
    p = chebfun.interp1(x, ff(x)); % INTERPOLAZIONE NEI NODI x.
    fc = chebfun(ff, n+1); % INTERPOLAZIONE IN (n+1) NODI DI
        CHEBYSHEV.
    eequi=[eequi norm(f-p,inf)];
    echeb=[echeb norm(f-fc,inf)];
end
clf;
semilogy(nn,eequi,'r-'); pause; semilogy(nn,echeb,'r-');
```

e di seguito

```
>> [eequi,echeb]=esempio1;
```

Esempio 1. Interpolazione in nodi equispaziati.

Quale risultato abbiamo le seguenti figure.

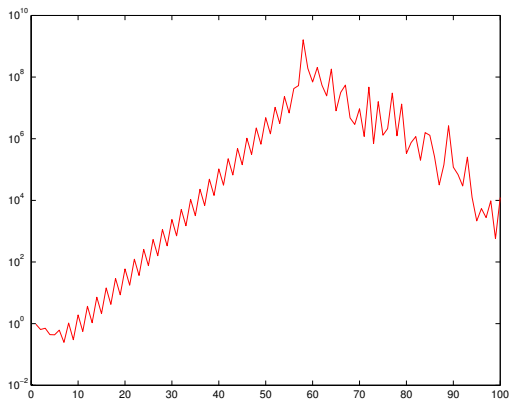


Figura: Grafico in scala semilogaritmica degli errori delle interpolanti in nodi equispaziati, al crescere del numero di nodi.

Esempio 1. Interpolazione in nodi equispaziati.

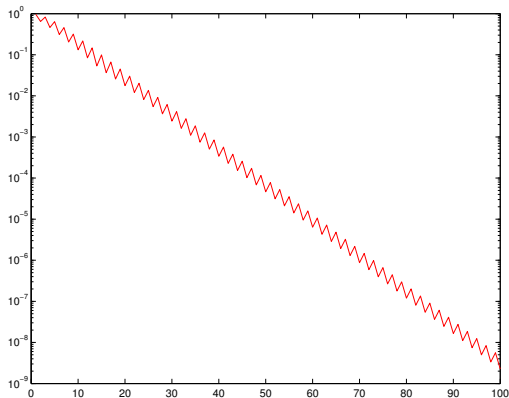


Figura: Grafico in scala semilogaritmica degli errori delle interpolanti in nodi di Chebyshev al crescere del numero di nodi.

Esempio 1. Interpolazione in nodi equispaziati.

I grafici mostrano che

- l'interpolazione in $n + 1$ nodi equispaziati fornisce polinomi p_n che non convergono alla funzione di Runge f ;
- l'interpolazione in $n + 1$ nodi di Chebyshev fornisce polinomi p_n che convergono alla funzione di Runge f .

Dal punto di vista della programmazione si noti che

- nella chiamata `f=chebfun(ff,d)`; si può dire in quale intervallo approssimare ff tramite funzioni polinomiali a tratti di tipo `chebfun`.
- nella chiamata `fc = chebfun(ff, n+1)`; si determina il polinomio di Chebyshev di grado n che interpola ff .
- nella chiamata `norm(f-p,inf)`; si approssima l'errore in norma infinito tra f e p .

Esercizio (1, facoltativo)

Sapendo che l'errore in norma infinito dell'interpolante nei nodi di Chebyshev è asintoticamente del tipo $C\gamma^n$, determinare C e γ .

Suggerimento:

- Calcolare il rapporto tra $e_{100} \approx C\gamma^{100}$ e $e_{98} \approx C\gamma^{98}$ e dedurre γ .
- Nota un'approssimazione di γ , calcolare C da $C \approx e_{100}/\gamma^{100}$.

Nota.

La correzione di questo esercizio è visibile nel file [esercizio1](#).

Esercizio (2, Facoltativo)

Mediante Chebfun, si approssimino in $[-1, 1]$

1 $f(x) = |x - 0.3|;$

2 $f(x) = \exp(x^2);$

3 $f(x) = \exp(x);$

4 $f(x) = \sin(x);$

5 $f(x) = \text{sinc}(x)$ dove $\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$ se $x \neq 0$, 1 se $x = 0$.

- Serve 'splitting', 'on'?
- Qual'è il numero di punti di Chebyshev affinché la funzione sia approssimata alla precisione di macchina in ogni esempio?
- La funzione sinc è predefinita, se installato il Signal Processing toolbox. Lo è pure in Chebfun, ma risulta $\text{sinc}(x) = \frac{\sin(x)}{x}$. Nel nostro esempio, definire:


```
f0=chebfun('pi*x'); f=sinc(f0);
```

 In alternativa, definire


```
f=@(x) sin(pi*x)./(pi*x+(x == 0));
```

Esempio 2. Miglior approssimante in norma infinito. Facoltativo.

Il comando `minimax` di una funzione f definita tramite `chebfun` fornisce la miglior approssimante in norma infinito di f (algoritmo di Remez). La routine `remez` è utilizzabile ma non suggerita. Vediamo la routine [esempio2.m](#).

```
function esempio2
d = [-1, 1]; % INTERVALLO.
ff = @(x) 1./(1+25*x.^2);
f=chebfun(ff,d, 'splitting', 'on');
n=10;
p = minimax(f,n); % MIGLIOR APPROX GRADO n
fc = chebfun(ff, n+1); % INTP. CHEB.
erem=norm(f-p,inf); echeb=norm(f-fc,inf);
fprintf('\n\t REMEZ: %1.5e',erem);
fprintf('\n\t CHEBYSHEV: %1.5e',echeb);
```

Nota.

Sia Λ_{10} la costante di Lebesgue nei nodi di Chebyshev $\{x_k\}_{k=0,\dots,10}$ utili per interpolare a grado 10. Da $2.42 \leq \Lambda_{10} \leq 2.43$, detto p_{10} il polinomio interpolatore le coppie $\{(x_k, f(x_k))\}_{k=0,\dots,10}$, ricaviamo

$$\|f - p_{10}\|_{\infty} \leq (1 + \Lambda_{10})E_{10}(f) \leq 3.43 \cdot E_{10}(f)$$

dove l'errore di miglior approssimazione $E_{10}(f)$ è numericamente quello compiuto dal polinomio fornito dall'algoritmo di Remez, a grado 10.

Digitiamo quindi in workspace

```
>> esempio2

REMEZ: 6.59229e-02
CHEBYSHEV: 1.32197e-01

>> % (1+lambda_n)E_n(f) e' circa 3.43*6.59229e-02, ovvero
    circa 2.261e-01.
```

- Come previsto la miglior approssimante di grado n offre risultati migliori dell'interpolante in $n + 1$ nodi di Chebyshev.
- D'altro canto la differenza non è molta, relativamente agli errori assoluti compiuti.
- Si noti che nella chiamata "f=chebfun(ff,d,'splitting','on');" si può dire in quale intervallo approssimare ff tramite funzioni polinomiali a tratti.
- Si noti che nella chiamata $fc = chebfun(ff, n+1);$ si determina il polinomio di Chebyshev di grado n che interpola ff .

Esercizio (3)

Si modifichi la routine `esempio1.m` utilizzando quanto visto in `esempio2.m`, così da paragonare la miglior approssimante di grado $n = 10, 20, \dots, 50$ con l'interpolante in $n + 1$ nodi di Chebyshev, sulle funzioni definite in $[-1, 1]$:

1 $f(x) = |x - 0.3|;$

2 $f(x) = \exp(x^2);$

3 $f(x) = \exp(x);$

4 $f(x) = \sin(x);$

5 $f(x) = \text{sinc}(x)$ dove $\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$ se $x \neq 0$, 1 se $x = 0$.

- Serve 'splitting', 'on' (chiedersi se la domanda ha senso)?
- Negli esempi forniti è molta la differenza tra l'usare l'interpolante in nodi di Chebyshev e la miglior approssimante?

Nota: In caso di necessità quale funzione sinc usare

```
f=@(x) sin(pi*x)./(pi*x+(x == 0))+(x == 0);
```

Esempio 3. Costanti di Lebesgue. Facoltativo.

Dalla teoria è noto che asintoticamente la crescita della costante di Lebesgue in nodi di

- Chebyshev, unisolventi a grado n è

$$\Lambda_n \sim \frac{2}{\pi} \left(\log(n+1) + \gamma + \log\left(\frac{8}{\pi}\right) \right)$$

dove

$$\gamma = 0.5772156649015328606065120900824 \dots$$

è la costante di Eulero-Mascheroni.

- nodi equispaziati, unisolventi a grado n è

$$\Lambda_n \sim \frac{2^{n+1}}{\exp(1) n \log(n)}$$

Esempio 3. Costanti di Lebesgue. Facoltativo.

Lo verifichiamo tramite un esempio, salvato in [esempio3.m](#)

```
function esempio3

fprintf('\n \t COSTANTI DI LEBESGUE, NODI CHEBSHEV');
max_deg=50;
em=0.57721566490153286060651209008240243104215933593992; %EULERO-MASCHERONI
for deg=1:max_deg
    x=chebpts(deg+1); % PUNTI DI CHEB. IN CHEBFUN. PS: length(x)=deg+1.
    [L,Lconst]=lebesgue(x); % FUNZIONE E COSTANTE DI LEBESGUE.
    lebesgue_constants(deg)=Lconst;
    est(deg)=(2/pi)*(log(deg+1)+em+log(8/pi)); % STIMA COSTANTE LEBESGUE.
    fprintf('\n \t DEG: %3.0f LEBESGUE CONST.: %1.4e EST.:% 1.4e' ,...
        deg,Lconst,est(deg));
end

plot(1:max_deg,lebesgue_constants,'ro-');
```

Come risultato otteniamo

```
COSTANTI DI LEBESGUE, NODI CHEBYSHEV
DEG:  1 LEBESGUE CONST.: 1.0000e+00 EST.: 1.4038e+00
DEG:  2 LEBESGUE CONST.: 1.2500e+00 EST.: 1.6619e+00
DEG:  3 LEBESGUE CONST.: 1.6667e+00 EST.: 1.8451e+00
DEG:  4 LEBESGUE CONST.: 1.7988e+00 EST.: 1.9871e+00
DEG:  5 LEBESGUE CONST.: 1.9889e+00 EST.: 2.1032e+00
DEG:  6 LEBESGUE CONST.: 2.0826e+00 EST.: 2.2013e+00
DEG:  7 LEBESGUE CONST.: 2.2022e+00 EST.: 2.2863e+00
DEG:  8 LEBESGUE CONST.: 2.2747e+00 EST.: 2.3613e+00
DEG:  9 LEBESGUE CONST.: 2.3619e+00 EST.: 2.4284e+00
DEG: 10 LEBESGUE CONST.: 2.4210e+00 EST.: 2.4891e+00
... ..
DEG: 40 LEBESGUE CONST.: 3.3105e+00 EST.: 3.3267e+00
DEG: 41 LEBESGUE CONST.: 3.3267e+00 EST.: 3.3420e+00
DEG: 42 LEBESGUE CONST.: 3.3416e+00 EST.: 3.3570e+00
DEG: 43 LEBESGUE CONST.: 3.3570e+00 EST.: 3.3716e+00
DEG: 44 LEBESGUE CONST.: 3.3712e+00 EST.: 3.3859e+00
DEG: 45 LEBESGUE CONST.: 3.3859e+00 EST.: 3.3999e+00
DEG: 46 LEBESGUE CONST.: 3.3996e+00 EST.: 3.4136e+00
DEG: 47 LEBESGUE CONST.: 3.4136e+00 EST.: 3.4270e+00
DEG: 48 LEBESGUE CONST.: 3.4267e+00 EST.: 3.4401e+00
DEG: 49 LEBESGUE CONST.: 3.4402e+00 EST.: 3.4530e+00
DEG: 50 LEBESGUE CONST.: 3.4527e+00 EST.: 3.4656e+00
```

Esempio 3. Costanti di Lebesgue.

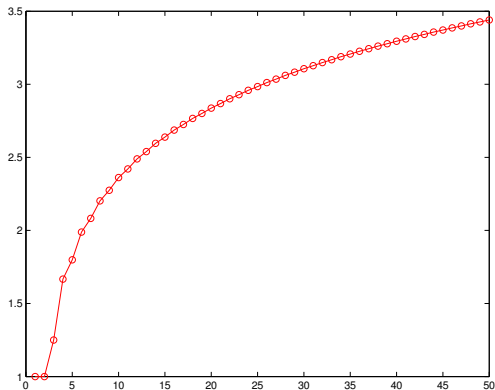


Figura: Grafico della costante di Lebesgue dei nodi di Chebyshev, per gradi compresi tra 1 e 50.

Esercizio (4 (Facoltativo))

Verificare che per nodi equispaziati, unisolventi a grado n , per $n = 1, \dots, 50$ si ha che

$$\frac{2^{n-2}}{n^2} \leq \Lambda_n \leq \frac{2^{n+3}}{n}$$

come stabilito in [Two results on Polynomial Interpolation in Equally Spaced Points](#), da Trefethen e Weideman, *J.A.T. (65)*, 247-260 (1991).

E' buona la stima asintotica $\Lambda_n \sim \frac{2^{n+1}}{\exp(1)n \log(n)}$?

Suggerimento:

- Modificare l'esempio 3,
- ricordare il comando `linspace`.

Approssimazione con polinomi trigonometrici e Chebfun.

Dalla release di Chebfun 5, Chebfun è in grado di trattare polinomi trigonometrici invece di algebrici, come descritto nella guida [Periodic Chebfuns](#).

Si approssima una generica funzione “ u ” (di default in $[-\pi, \pi]$), tramite

$$q_N(t) = \begin{cases} \sum_{k=-(N-1)/2}^{(N-1)/2} c_k \exp(ikt), & \text{se } N \text{ dispari,} \\ \sum_{k=-N/2}^{N/2} c_k \exp(ikt) + c_{N/2} \cos(\frac{N}{2}t), & \text{se } N \text{ pari,} \end{cases}$$

dove

$$c_k := \frac{1}{N} \sum_{j=0}^{N-1} u(t_j) \exp(-ikt_j), \quad t_j = -\pi + 2\pi j/N.$$

con c_k calcolati in $O(N \log(N))$ operazioni con la **Fast Fourier Transform** (cf. [4, p.557]).

Quale esempio trattiamo la funzione $\tanh(3 \sin(t)) - \sin(t + \frac{1}{2})$.

```
>>> f=chebfun(@(t) tanh(3*sin(t))-sin(t+1/2), [-pi pi])
f =
    chebfun column (1 smooth piece)
      interval      length      endpoint values
 [   -3.1,    3.1]    226      0.48      0.48
vertical scale = 0.92

>>> format long e
>>> f(-pi)
ans =
    4.794255386042027e-01
>>> f(pi)
ans =
    4.794255386042033e-01
>>> plot(f, 'LineWidth', 1.6)
```

E' stato quindi necessario determinare un certo **polinomio algebrico** di grado 225 (interpolante 226 punti di Chebyshev), per avere un errore dell'ordine della precisione di macchina.

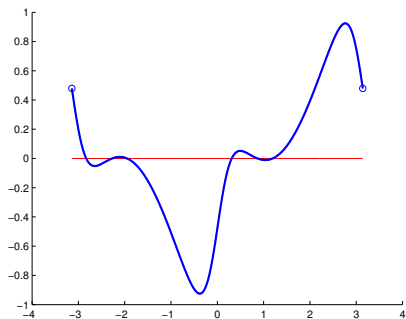


Figura: Grafico della funzione periodica $u(x) = \tanh(3 \sin(t)) - \sin(t + \frac{1}{2})$ in cui $u(-\pi) = u(\pi) \approx 4.794255386042033e - 01$.

Approssimiamo ora con polinomi trigonometrici, aggiungendo alla chiamata `chebfun`, la preferenza `'trig'`.

```
>> f=chebfun(@(t) tanh(3*sin(t))-sin(t+1/2), [-pi pi], 'trig')
f =
    chebfun column (1 smooth piece)
      interval      length  endpoint  values  trig
 [   -3.1,    3.1]    147      0.48    0.48
vertical scale = 0.92
>>
```

E' stato quindi necessario determinare un certo polinomio:

- **trigonometrico** interpolante la funzione in 147 punti equispaziati di $[-\pi, \pi]$, per avere un errore dell'ordine della precisione di macchina,
- **algebrico** interpolante la funzione in 226 punti di Chebyshev in $[-\pi, \pi]$.

In effetti $147 \cdot \pi/2 = 230.9071 \approx 226$.

Nota.

Si noti che per funzioni periodiche regolari, questo fattore $\pi/2$ deriva dal fatto che

- *interpolanti trigonometriche hanno un potere risolutivo di 2 punti per lunghezza d'onda,*
- *le interpolanti di Chebyshev hanno un potere risolutivo di π punti per lunghezza d'onda,*

che tradotto altrimenti dice che con polinomi trigonometrici ci si aspetta meno campionamenti rispetto alle classiche chebfun.

Cambiamo funzione, e consideriamo $f(x) = 5 \sin(3t) + 6 \cos(2t)$.

```
>>> f=chebfun(@(t) 5*sin(3*t)+6*cos(2*t), [-pi pi], 'trig')
f =
    chebfun column (1 smooth piece)
      interval      length      endpoint values trig
 [   -3.1,    3.1]      7          6          6
vertical scale = 9.3
>>>
```

Viene da domandarsi quale sia il polinomio trigonometrico ottenuto. Essendo facilmente dall'identità di Eulero

$$\sin(kt) = \frac{i \cdot (\exp(-ikt) - \exp(ikt))}{2}, \quad \cos(kt) = \frac{\exp(ikt) + \exp(-ikt)}{2}$$

abbiamo che

$$\begin{aligned} f(t) &= 5 \sin(3 \cdot t) + 6 \cos(2 \cdot t) \\ &= 5i \cdot \frac{\exp(-i3t) - \exp(i3t)}{2} + 6 \cdot \frac{\exp(i2t) + \exp(-i2t)}{2} \\ &= (5i/2) \exp(-i3t) + 3 \exp(-i2t) + 3 \exp(i2t) - (5i/2) \exp(i3t). \end{aligned}$$

Approssimazione con polinomi trigonometrici e Chebfun.

Utilizzando il comando `trigcoeffs`, che determina i coefficienti di Fourier $c_k^* = c_{-M-1+k}$ (attenzione all'ordine!), con $M = (N - 1)/2$ se N dispari o $M = N/2$ se N pari,

```
>> trigcoeffs(f)
ans =
    4.762103912697982e-18 + 2.500000000000000e+00i
    2.999999999999999e+00 - 4.598694340586186e-17i
    7.446384377092827e-16 + 1.360023205165817e-15i
    7.216449660063518e-16 + 0.000000000000000e+00i
    7.446384377092827e-16 - 1.360023205165817e-15i
    2.999999999999999e+00 + 4.598694340586186e-17i
    4.762103912697982e-18 - 2.500000000000000e+00i
>>
```

e quindi i coefficienti sono, tolte le quantità quasi nulle,

$$[2.5i, 3, 0, 0, 0, 3, -2.5i]$$

come richiesto, in quanto

$$f(t) = (5i/2) \exp(-i3t) + 3 \exp(-i2t) + 3 \exp(i2t) - (5i/2) \exp(i3t).$$

- Utilizzando un comando del tipo

```
f1=chebfun(@(t) -sign(abs(t)-pi/2),[-pi pi], 'trunc',N, 'trig');
```

si calcola una **certa approssimazione** di una particolare **onda quadra** mediante serie trigonometriche troncate con **N coefficienti** (cf. [2]).

- Utilizzando un comando del tipo

```
f1=chebfun(@(t) -sign(abs(t)-pi/2),[-pi pi],N, 'trig');
```

si calcola l'**interpolante polinomiale** trigonometrica di grado **$(N-1)/2$** , mediante FFT (cf. [2]).

In generale useremo valori N dispari, anche se Chebfun permette N pari.

Esempio 4. Fenomeno di Gibbs.

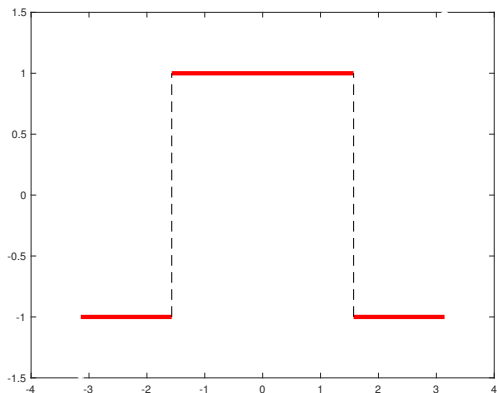


Figura: Grafico della funzione discontinua $-\text{sign}(|t| - \pi/2)$ in $[-\pi, \pi]$.

Esempio 4. Fenomeno di Gibbs.

Si consideri la funzione $-\text{sign}(\text{abs}(t)-\pi/2)$ e la si approssimi con polinomi trigonometrici. Utilizziamo il codice `esempio4.m`

```
function esempio4
warning off;
N1=11;f1=chebfun(@(t) -sign(abs(t)-pi/2),[-pi pi], 'trunc',N1, 'trig');maxN1=max(f1);
N2=31;f2=chebfun(@(t) -sign(abs(t)-pi/2),[-pi pi], 'trunc',N2, 'trig');maxN2=max(f2);
N3=5001;f3=chebfun(@(t) -sign(abs(t)-pi/2),[-pi pi], 'trunc',N3, 'trig');maxN3=max(f3);
fprintf('\n \t GIBBS SIZE (%3.0f): %1.5e',N1,maxN1)
fprintf('\n \t GIBBS SIZE (%3.0f): %1.5e',N2,maxN2)
fprintf('\n \t GIBBS SIZE (%3.0f): %1.5e',N3,maxN3)
hold on;
xx=linspace(-pi,pi,1000); yy=-sign(abs(xx)-pi/2);
clf; hold on; plot(xx,yy, 'r-'); plot(f1, 'k-', 'LineWidth', 2); hold off;
fprintf('\n \t PAUSE'); pause;
clf; hold on; plot(xx,yy, 'r-'); plot(f2, 'k-', 'LineWidth', 2);
hold off; fprintf('\n \t PAUSE'); pause;
clf; hold on; plot(xx,yy, 'r-'); plot(f3, 'k-', 'LineWidth', 2); hold off;
fprintf('\n \t PAUSE'); pause;
fprintf('\n \n');
```

Nota.

Si osservi che in GIBBS SIZE mostriamo il massimo valore della approssimante, da paragonare col massimo valore dell'onda quadra che è 1.

Otteniamo

```
>> esempio4  
  
GIBBS SIZE ( 11): 1.18836e+00  
GIBBS SIZE ( 31): 1.18028e+00  
GIBBS SIZE (5001): 1.17801e+00  
PAUSE  
PAUSE  
PAUSE  
  
>>
```

e tre grafici con delle gobbe sorprendenti, circa della stessa altezza (appunto $1.18836e + 00$, $1.18028e + 00$ e $1.17801e + 00$ per interpolanti trigonometriche in cui $N = 11$, $N = 31$, $N = 51$).

Nota.

Si osservi che in questo codice si calcola una certa approssimazione mediante serie trigonometriche troncate con N coefficienti e non l'interpolante polinomiale trigonometrica di grado $(N - 1)/2$.

Esempio 4. Fenomeno di Gibbs.

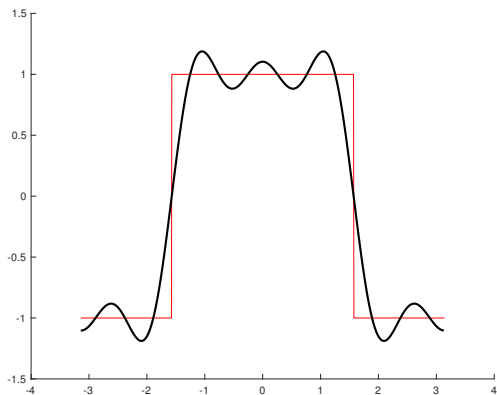


Figura: Grafico della funzione $-sign(|t| - \pi/2)$ in $[-\pi, \pi]$ e approssimante trigonometrica con $N = 11$.

Esempio 4. Fenomeno di Gibbs.

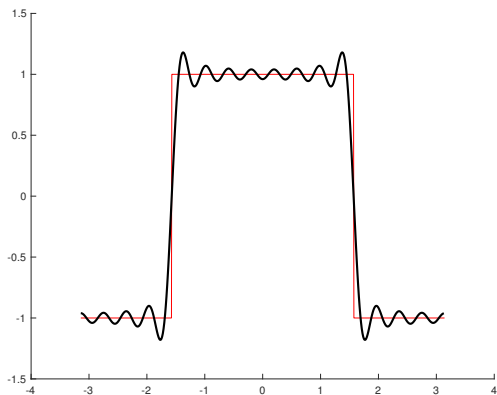


Figura: Grafico della funzione $-\text{sign}(|t| - \pi/2)$ in $[-\pi, \pi]$ e approssimante trigonometrica con $N = 31$.

Esempio 4. Fenomeno di Gibbs.

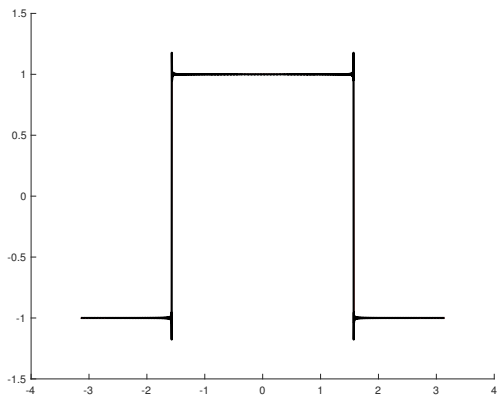


Figura: Grafico della funzione $-\text{sign}(|t| - \pi/2)$ in $[-\pi, \pi]$ e approssimante trigonometrica con $N = 5001$.

Questo fenomeno, detto **di Gibbs** (1899) anche se scoperto nel 1848 da Wilbraham, mostra un problema nell'approssimare mediante serie di Fourier funzioni con salti, pur essendo in $L_2(-\pi, \pi)$.

Problema:

- se così fosse, al variare dei nodi, si può avere la convergenza in norma infinito?
- se così fosse, al variare dei nodi, si può avere la convergenza in norma 2?

Teorema

Se $f : \mathbb{R} \rightarrow \mathbb{R}$ è continua a tratti e differenziabile, di periodo L

$$S_N f(x) = \sum_{k=-N}^N a_k \exp\left(\frac{2\pi i N k x}{L}\right)$$

e in x_0 è presente il salto $a = f(x_0^+) - f(x_0^-) \neq 0$ allora

- $\lim_N S_N f(x_0^+ + \frac{L}{2N}) = f(x_0^+) + a \cdot 0.089490 \dots$
- $\lim_N S_N f(x_0^- - \frac{L}{2N}) = f(x_0^-) - a \cdot 0.089490 \dots$

Si noti che

- nei punti $x_0^+ + \frac{L}{2N}$, $x_0^- - \frac{L}{2N}$ avvengono le *gobbe* del fenomeno di Gibbs;
- nel caso di N dispari, la chiamata Chebfun calcola $\sum_{k=-M}^M c_k \exp(ikt)$, con $M = (N - 1)/2$.

Questo teorema ci dice che in x_0 la funzione ha un salto di ampiezza

$$a = f(x_0^+) - f(x_0^-) \neq 0$$

allora asintoticamente in N la serie di Fourier presenterà

- in $x_0^+ + \frac{L}{2N}$ valore circa $f(x_0^+) + a \cdot 0.089490$;
- in $x_0^+ - \frac{L}{2N}$ valore circa $f(x_0^-) - a \cdot 0.089490$.

Nel nostro caso $a = 2$, $x_0 = \pm\pi/2$, e quindi l'errore assoluto compiuto è approssimativamente, e asintoticamente, tanto *leggermente* a sinistra quanto *leggermente* a destra di ogni discontinuità

$$2 \cdot 0.089490 \approx 0.1790$$

Siccome l'onda quadra é facilmente in $L_2(-\pi.\pi)$, dalla teoria sappiamo le proprietá di convergenza dell'approssimante in norma L_2 .

In effetti l'errore compiuto risulta come da questa tabella:

N	$\ f - s_N\ _\infty$	$\ f - s_N\ _2$
11	1.2e+00	6.5e-01
31	1.2e+00	4.0e-01
5001	1.2e+00	2.9e-02
10001	1.2e+00	1.8e-02
20001	1.2e+00	1.0e-02
40001	1.2e+00	3.3e-03

Esercizio 4. Fenomeno di Gibbs.

Esercizio (4)

Riprovare l'esempio precedente utilizzando l'interpolante trigonometrica polinomiale di grado $N = 11, 31, 5001$.

- Si noti che ciò implica assegnare $2*N+1, 'trig'$ invece di $'trunc', N, 'trig'$;
- per completezza, la valutiamo nei punti $\frac{-\pi}{2} + \frac{L}{2N} = \frac{-\pi}{2} + \frac{\pi}{N}$, essendo $L = 2\pi$.

Svolgimento.

Dal codice

```
function esercizio4
N=[11 31 5001]; f=@(t) -sign(abs(t)-pi/2);
for k=1:length(N)
    n=N(k); ftrig=chebfun(f,[-pi pi],2*n+1, 'trig');
    val=-pi/2+pi/n; fprintf('\n \t GIBBS SIZE (%3.0f): %1.5e \n ',n,ftrig(val))
end
```

scriviamo su shell

```
>> esercizio4
GIBBS SIZE ( 11): 1.15961e+00
GIBBS SIZE ( 31): 1.16206e+00
GIBBS SIZE (5001): 1.16132e+00
>>
```

Le quantità ottenute molto prossime ai valori previsti dal teorema sopracitato.

Esercizio (5)

Cosa succede se invece di polinomi trigonometrici, si approssima la funzione

$$f(t) = -\text{sign}(|t| - \pi/2)$$

in $[-\pi, \pi]$ con il polinomio algebrico interpolante in nodi di Chebyshev scalati, a grado 5, 30, 5000 via Chebfun?

Suggerimento:

- 1 utilizzare quanto fatto in un esercizio precedente (anche relativamente all'intervallo da usare!),*
- 2 osservare quanti nodi bisogna richiedere per avere una interpolante di grado richiesto.*

La soluzione al problema é disponibile nel file [esercizio5_correzione.m](#)

Dalla teoria sappiamo che se i $2n + 1$ coefficienti di Fourier in $[0, 2\pi]$ sono approssimati mediante la formula dei trapezi, allora si ottiene la interpolante polinomiale nei punti $t_j = j \frac{2\pi}{2n+1}$, per $j = 0, \dots, 2n$ (cf. [1, p.178]). Vediamo quale esempio la funzione [esempio5.m](#)

```
function esempio5
clear all; clf;
n=2; % GRADO INTP. TRIG.
f=@(x) 1./(1+(x-pi).^2); % FUNZ. DI RUNGE SHIFTATA IN [0,2*pi] (
    PERIODICA).
fa=chebfun(f,[0,2*pi]); % CHEBFUN ALGEBRICA.
ff=chebfun(f,[0 2*pi],2*n+1, 'trig'); % SERIE DI FOUR. (INTP).
% ATTENZIONE.
% IN ATKINSON SI DICE CHE INTERPOLA NEI PUNTI j*2*pi/(2*n+1), j
    =0,...,2*n.
t=linspace(0,2*pi,2*n+2); % NODI EQUISPAZIATI
AE=max(abs(f(t)-ff(t)));
fprintf('\n \t MAX. ERR. INTP. PTS.:%2.2e \n',AE);
plot(fa,'Linewidth',2); hold on; plot(ff,'Linewidth',2);
plot(t,ff(t),'o','MarkerEdgeColor','k','MarkerFaceColor','g','
    MarkerSize',8);
hold off;
legend('Funzione','Intp. Trig.');
```

Otteniamo

```
>> esempio5  
MAX. ERR. INTP. PTS.:1.11e-16  
>>
```

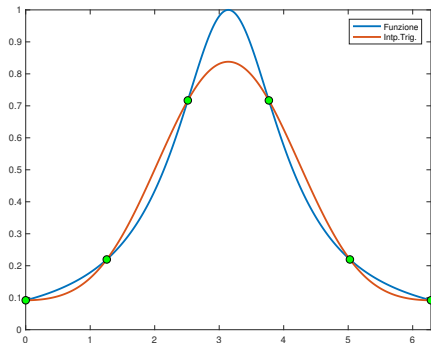


Figura: Grafico della funzione $-\text{sign}(|t| - \pi/2)$ in $[-\pi, \pi]$ e interpolante trigonometrica con $N = 5$.

Esercizio (Facoltativo)

Che errore si compie in norma ∞ e in norma 2, approssimando la funzione

$$f(t) = \log(0.001 + t), \quad t \in [0, 1]$$

con

- polinomi algebrici di grado 11, 31, 51 via Chebfun?
- polinomi trigonometrici di grado 11, 31, 51 via Chebfun (nella versione col 'trunc')?

Quale delle due approssimazioni è migliore? Ha senso quella trigonometrica (plot!)?

- Se deg è il grado allora 'trunc' va fatto per $N = \dots$
- Si noti che se fa è un oggetto chebfun di natura algebrica e ft di natura trigonometrica, allora si può fare $\text{norm}(fa-ft)$, $\text{norm}(fa-ft, \text{inf})$.
- Possibili warnings sorgono dalla versione trigonometrica di Chebfun.
- Versioni diverse di Matlab/Chebfun possono influenzare il risultato.

Nota.

Per avere qualche informazione su cosa succede analizzando funzioni non periodiche, per via trigonometrica, da

G. Wright, M. Javed, H. Montanelli, L.N. Trefethen, Extension of Chebfun to periodic functions, SIAM Sc. Comput., Vol. 37, No. 5, (2015) pp. 554-573:

If one tries to construct a trigfun by sampling a function that is not smoothly periodic, Chebfun will by default go up to length 2^{16} and then issue a warning:

```
>> h = chebfun('exp(t)', [0 2*pi], 'trig')  
Warning: Function not resolved using 65536 pts.  
Have you tried a non-trig representation?
```

On the other hand, computations that are known to break periodicity or smoothness will result in the representation being automatically cast from a trigfun to a chebfun.



K. Atkinson, *An Introduction to Numerical Analysis*, Wiley, 1989.



Periodic Chebfuns, <https://www.chebfun.org/docs/guide/guide11.html>.



A. Quarteroni, R. Sacco e F. Saleri *Matematica Numerica*, Springer, 1998.



G.B. Wright, M. Javed, H. Montanelli, L.N. Trefethen, *Extension of Chebfun to periodic functions*, SIAM J. Sci. Comp., 2015.



Wikipedia, (Fourier Series), http://en.wikipedia.org/wiki/Fourier_series.