

Chebfun

Alvise Sommariva

Università degli Studi di Padova
Dipartimento di Matematica

14 marzo 2017

Introduzione a Chebfun

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.

Per una guida si consulti

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

mentre per installare l'ambiente in Matlab, si veda

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

Introduzione a Chebfun

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} .

Introduzione a Chebfun

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 .

Introduzione a Chebfun

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

```
>> x=chebfun('x'); % Inizializzazione.  
>> f=1./(1+25*x.^2); % RUNGE.  
>> f  
f =  
  
    chebfun column (1 smooth piece)  
      interval      length endpoint values  
[      -1,      1]      181      0.038      0.038  
vertical scale = 1  
>>
```

La funzione di Runge è stata approssimata con un errore relativo di 10^{-15} utilizzando 181 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);
    p = chebfun.interpl(x, ff(x)); % INTERPOLAZIONE NEI NODI x.
    fc = chebfun(ff, n+1);
    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.

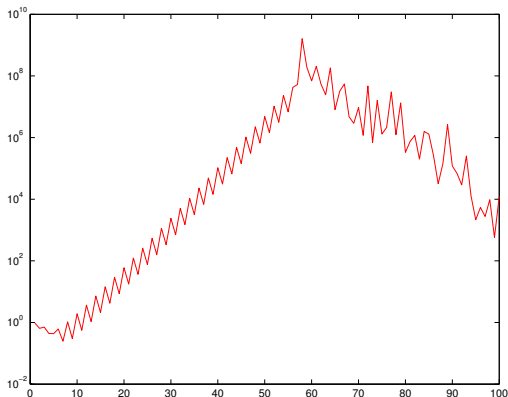


Figura : Grafico in scala semilogaritmica degli errori delle interpolanti in nodi equispaziati, al crescere del numero di nodi.

Esempio 1. Interpolazione in nodi equispaziati.

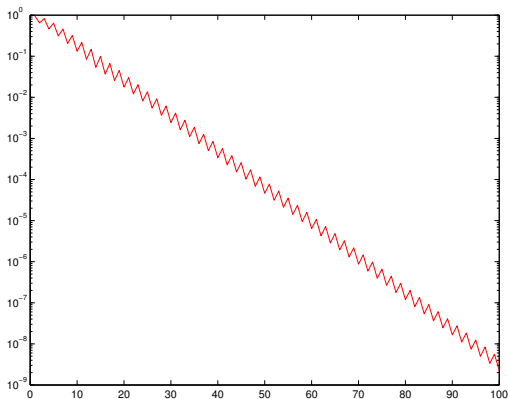


Figura : Grafico in scala semilogaritmica degli errori delle interpolanti in nodi di Chebyshev al crescere del numero di nodi.

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.

Esercizio (1)

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 $e_n \approx C\gamma^n$ e $e_{n-2} \approx C\gamma^{n-2}$ e dedurre γ .*
- *Noto γ , calcolare C .*

Esercizio 2.

Esercizio (2)

Si modifichi la precedente routine così da studiare l'approssimazione, similmente all'esempio precedente, le funzioni

- 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 $\text{sinc}(x) = \frac{\sin(x)}{x}$ se $x \neq 0$, 1 se $x = 0$ (funzione predefinita in `chebfun`).
- Serve 'splitting', 'on'?
 - Qual'e' il numero di punti di Chebyshev affinché la funzione sia approssimata alla precisione di macchina in ogni esempio?

Esempio 2. Miglior approssimante in norma infinito.

Il comando `remez` di una funzione f definita tramite `chebfun` fornisce la miglior approssimante in norma infinito di f . Vediamo un esempio, salvato in [esempio2.m](#).

```
function esempio2

d = [-1, 1]; % INTERVALLO.
ff = @(x) 1./(1+25*x.^2);
f=chebfun(ff,d,'splitting','on');
n=10;
p = remez(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);
```

Esempio 2. Miglior approssimante in norma infinito.

Digitiamo quindi in workspace

```
>> esempio2;  
  
REMEZ: 6.59229e-02  
CHEBYSHEV: 1.32197e-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.

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, relativamente alle funzioni

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 $\text{sinc}(x) = \frac{\sin(x)}{x}$ se $x \neq 0$, 1 se $x = 0$
(funzione predefinita in chebfun).

Esercizio 3.

- Serve 'splitting', 'on'?
- *Negli esempi forniti è molta la differenza tra l'usare l'interpolante in nodi di Chebyshev e la miglior approssimante?*

Esempio 3. Costanti di Lebesgue.

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

- Chebyshev, unisolventi a grado n è

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

dove $\gamma =$

0.57721566490153286060651209008240243104215933593992

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.

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

```
function esempio3

fprintf('\n \t COSTANTI DI LEBESGUE, NODI CHEBYSHEV');
max_deg=50;
em=0.57721566490153286060651209008240243104215933593992; %EULERO-MASCHERONI
for deg=1:max_deg
    x=chebpts(deg); % PUNTI DI CHEB. IN CHEBFUN. PS: length(x)=deg.
    [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-');
```

Esempio 3. Costanti di Lebesgue.

Come risultato otteniamo

```
COSTANTI DI LEBESGUE, NODI CHEBYSHEV
DEG:  1 LEBESGUE CONST.: 1.0000e+00 EST.: 1.4038e+00
DEG:  2 LEBESGUE CONST.: 1.0000e+00 EST.: 1.6619e+00
DEG:  3 LEBESGUE CONST.: 1.2500e+00 EST.: 1.8451e+00
DEG:  4 LEBESGUE CONST.: 1.6667e+00 EST.: 1.9871e+00
DEG:  5 LEBESGUE CONST.: 1.7988e+00 EST.: 2.1032e+00
DEG:  6 LEBESGUE CONST.: 1.9889e+00 EST.: 2.2013e+00
DEG:  7 LEBESGUE CONST.: 2.0826e+00 EST.: 2.2863e+00
DEG:  8 LEBESGUE CONST.: 2.2022e+00 EST.: 2.3613e+00
DEG:  9 LEBESGUE CONST.: 2.2747e+00 EST.: 2.4284e+00
DEG: 10 LEBESGUE CONST.: 2.3619e+00 EST.: 2.4891e+00
DEG: 11 LEBESGUE CONST.: 2.4210e+00 EST.: 2.5445e+00
DEG: 12 LEBESGUE CONST.: 2.4894e+00 EST.: 2.5954e+00
```

.

```
DEG: 40 LEBESGUE CONST.: 3.2948e+00 EST.: 3.3267e+00
DEG: 41 LEBESGUE CONST.: 3.3105e+00 EST.: 3.3420e+00
DEG: 42 LEBESGUE CONST.: 3.3267e+00 EST.: 3.3570e+00
DEG: 43 LEBESGUE CONST.: 3.3416e+00 EST.: 3.3716e+00
DEG: 44 LEBESGUE CONST.: 3.3570e+00 EST.: 3.3859e+00
DEG: 45 LEBESGUE CONST.: 3.3712e+00 EST.: 3.3999e+00
DEG: 46 LEBESGUE CONST.: 3.3859e+00 EST.: 3.4136e+00
DEG: 47 LEBESGUE CONST.: 3.3996e+00 EST.: 3.4270e+00
DEG: 48 LEBESGUE CONST.: 3.4136e+00 EST.: 3.4401e+00
DEG: 49 LEBESGUE CONST.: 3.4267e+00 EST.: 3.4530e+00
DEG: 50 LEBESGUE CONST.: 3.4402e+00 EST.: 3.4656e+00
```

Esempio 3. Costanti di Lebesgue.

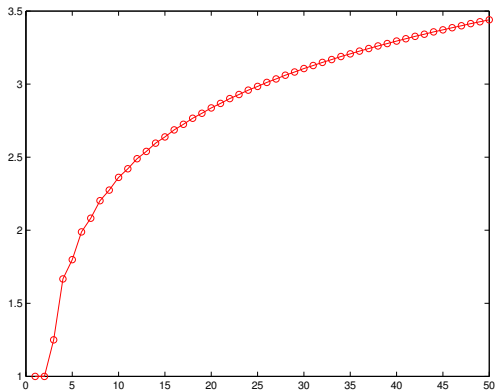


Figura : Grafico della costante di Lebesgue dei nodi di Chebyshev, per gradi compresi tra 1 e 50.

Esercizio 4.

Esercizio (4)

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 Chebfun](#). Quindi si approssima una generica funzione “ u ” (di default in $[-\pi, \pi]$), tramite serie troncate

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

dove

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

con c_k calcolati in $O(N \log(N))$ operazioni con la **Fast Fourier Transform**.

Approssimazione con polinomi trigonometrici e Chebfun.

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.

Approssimazione con polinomi trigonometrici e Chebfun.

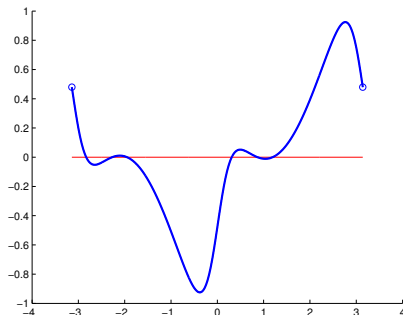


Figura : Grafico della funzione periodica $u(x) = \tanh(3 \sin(t)) - \sin(t + \frac{1}{2})$ in cui $u(-\pi) = u(\pi) \approx 4.794255386042033e - 01$.

Approssimazione con polinomi trigonometrici e Chebfun.

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, contro i 226 del polinomio algebrico precedentemente ottenuto. In effetti $147 * \pi/2 = 230.9071 \approx 226$.

Approssimazione con polinomi trigonometrici e Chebfun.

Nota.

Si noti che per funzioni periodiche regolari, questo 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.

Approssimazione con polinomi trigonometrici e 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

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

Approssimazione con polinomi trigonometrici e Chebfun.

Utilizzando il comando `trigcoeffs`, che determina i coefficienti di Fourier discreti $c_k^* = c_{-M-1+k}$, 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

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

Esempio 4. Fenomeno di Gibbs.

Si consideri la funzione $\text{sign}(x)$ e ci si proponga di approssimarla con polinomi trigonometrici. Utilizziamo il codice

```
function esempio4
warning off;
N1=10; f1=chebfun(@(t) -sign(abs(t)-pi/2),[-pi pi], 'trunc', N1, 'trig'); maxN1=
    max(f1);
N2=30; f2=chebfun(@(t) -sign(abs(t)-pi/2),[-pi pi], 'trunc', N2, 'trig'); maxN2=
    max(f2);
N3=5000; 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)

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');
```

Esempio 4. Fenomeno di Gibbs.

Otteniamo

```
>> esempio4  
  
GIBBS SIZE ( 10): 1.18836e+00  
GIBBS SIZE ( 30): 1.18028e+00  
GIBBS SIZE (5000): 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 $N = 10$, $N = 30$, $N = 50$).

Esempio 4. Fenomeno di Gibbs.

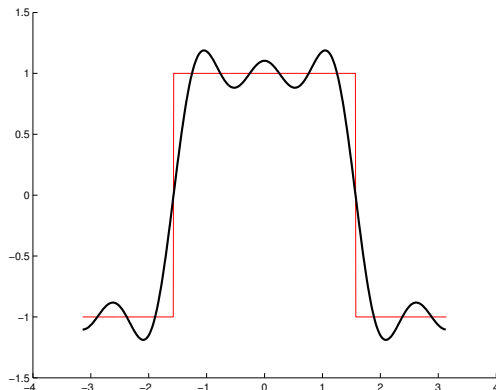


Figura : Grafico della funzione $-sign(|t| - \pi/2)$ in $[-\pi, \pi]$ e interpolante trigonometrica con $N = 10$.

Esempio 4. Fenomeno di Gibbs.

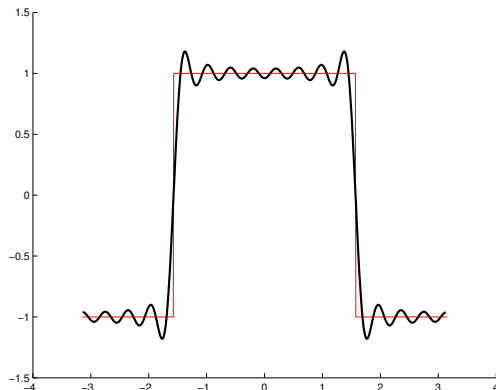


Figura : Grafico della funzione $-sign(|t| - \pi/2)$ in $[-\pi, \pi]$ e interpolante trigonometrica con $N = 30$.

Esempio 4. Fenomeno di Gibbs.

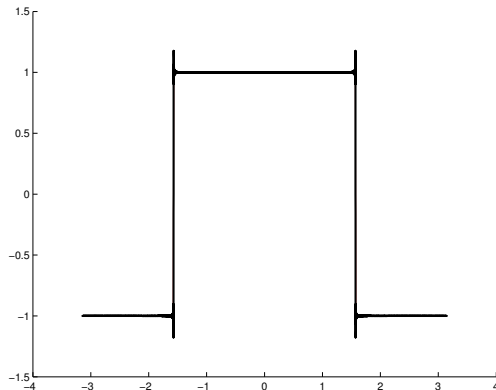


Figura : Grafico della funzione $-sign(|t| - \pi/2)$ in $[-\pi, \pi]$ e interpolante trigonometrica con $N = 5000$.

Esempio 4. Fenomeno di Gibbs.

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?

Esempio 4. Fenomeno di Gibbs.

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

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

e

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

allora

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

Si noti che nei punti $x_0^+ + \frac{L}{2N}$, $x_0^- - \frac{L}{2N}$ avvengono le *gobbe* del fenomeno di Gibbs.

Le quantità precedentemente calcolate sono dell'ordine predetto da questi teoremi?

Esercizio 5.

Esercizio (5)

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

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

in $[-\pi, \pi]$ con polinomi algebrici di grado 10, 30, 50 via Chebfun?

Esercizio Facoltativo.

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 10, 30, 50 via Chebfun?*
- *polinomi trigonometrici di grado 10, 30, 50 via Chebfun?*

Quale delle due approssimazioni è migliore? Ha senso quella trigonometrica per funzioni non periodiche?

Esercizio Facoltativo.

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.