Interpolazione trigonometrica e FFT

19 gennaio 2009

1 Interpolazione trigonometrica

Sia f una funzione continua e periodica nell'intervallo $[0, 2\pi]$ e si pongano

$$t_j = j \cdot \frac{2\pi}{2n+1},\ j = 0, \ldots, 2n.$$

Il problema dell'interpolazione trigonometrica consiste nel calcolare i coefficienti $\tilde{c}_k$, $k = -n, \ldots, n$ tali che

$$\sum_{k=-n}^{n} \tilde{c}_k \exp(-ikt_j) = f(t_j),\ j = 0, \ldots, 2n.$$

Osserviamo che

$$\sum_{j=0}^{2n} \exp(-i(k-l)t_j) = \begin{cases} 0 & k \neq l \\ 2n+1 & k = l \end{cases}$$

Il caso in cui $k = l$ è ovvio e quindi poniamo attenzione al caso in cui $k \neq l$. Si vede subito che posto $w = \exp(-i(k-l) \cdot \frac{2\pi}{2n+1})$

$$\begin{aligned}
\sum_{j=0}^{2n} \exp(-i(k-l)t_j) &= \sum_{j=0}^{2n} \exp(-i(k-l)j \cdot \frac{2\pi}{2n+1}) \\
&= \sum_{j=0}^{2n} w^j = (w^{2n+1} - 1)/(w-1) \\
&= \left(\exp\left(-i(k-l) \cdot \frac{2\pi}{2n+1}\right)^{2n+1} - 1\right)/(w-1) \\
&= \left(\exp(-i(k-l) \cdot 2\pi) - 1\right)/(w-1) \\
&= 0 \qquad (1)
\end{aligned}$$

Moltiplicando ambo i membri per $\exp(+ilt_j)$ e sommando in j, otteniamo

$$\begin{aligned}
\sum_{j=0}^{2n} f(t_j) \exp(+ilt_j) &= \sum_{j=0}^{2n} \exp(+ilt_j) \sum_{k=-n}^{n} \tilde{c}_k \exp(-ikt_j) \\
&= \sum_{k=-n}^{n} \tilde{c}_k \sum_{j=0}^{2n} \exp(-i(k-l)t_j) \\
&= (2n+1)\,\tilde{c}_l \qquad (2)
\end{aligned}$$

da cui

$$\tilde{c}_l = \frac{1}{2n+1} \sum_{j=0}^{2n} f(t_j) \exp(+i l t_j).$$

Una prima lettura non banale consiste nel fatto che i coefficienti c_k sono noti esplicitamente, senza risolvere numericamente il sistema lineare del tipo $Vc = f$ dove

$$V_{jk} = \exp(-ikt_j),\ f_j = f(t_j).$$

Osserviamo ora che in (**??**), per $k = -n, \dots, n$ avevamo approssimato numericamente c_k con la formula dei trapezi fornendo

$$c_k \approx \frac{1}{N} \sum_{s=1}^{N} f(t_{s-1}) \exp\left(+i\,k\,(s-1)\,\frac{2\pi}{N}\right) \tag{3}$$

Nel caso speciale in cui $N = 2n+1$, posto $j = s-1$, abbiamo essendo $t_j = j \cdot \frac{2\pi}{2n+1}$

$$c_k \approx \frac{1}{2n+1} \sum_{j=0}^{2n} f(t_j) \exp\left(+i\,k\,j\,\frac{2\pi}{2n+1}\right) = \tilde{c}_k \tag{4}$$

cioè l'interpolante polinomiale ha quali coefficienti $\tilde{c}_{k=-N}^{N}$ quelli ottenuti calcolando numericamente i coefficienti di Fourier c_k con una formula dei trapezi avente $2N+1$ punti.

2 FFT

Il problema consiste nel calcolare i coefficienti di Fourier $\{c_j\}_{j=0,\dots,N-1}$ per una funzione

$$f(x) = \sum_{j=0}^{N-1} c_j \exp(ijx)$$

i cui valori sono noti nei punti $\frac{2\pi\beta}{N}$ con $\beta = 0, 1, \dots, N-1$. Per quanto visto,

$$c_j = \frac{1}{N} \sum_{\beta=0}^{N-1} f\left(\frac{2\pi\beta}{N}\right) \exp\left(\frac{-ij2\pi\beta}{N}\right),\ j = 0, \dots, N-1. \tag{5}$$

Si ponga

$$w = \exp\left(\frac{2\pi i}{N}\right),\ a_\beta = \frac{1}{N} f\left(\frac{2\pi\beta}{N}\right)$$

Possiamo quindi riscrivere il problema come segue: per $j = 0, \dots, N-1$, calcolare

$$c_j = \sum_{\beta=0}^{N-1} a_\beta w^{j\beta} \text{ dove } w^N = 1 \tag{6}$$

Se per operazione intendiamo una moltiplicazione e una somma in campo complesso, allora usando lo schema di Horner, si può risolvere questo problema in N^2

operazioni. L'algoritmo noto come FFT è diventato popolare dopo uno studio di Cooley and Tukey del 1966 (cf. [2]), ma pare fosse già noto in forma limitata a Gauss (1805), permette di risolvere il problema con una complessità inferiore. Consideriamo il caso $N = 2^k$. Sia $\beta = 2\beta_1$ quando β è pari, e $\beta = 2\beta_1 + 1$ quando β è dispari. Da (6) abbiamo

$$c_j = \sum_{\beta_1=0}^{\frac{N}{2}-1} a_{2\beta_1}(w^2)^{j\beta_1} + \sum_{\beta_1=0}^{\frac{N}{2}-1} a_{2\beta_1+1}(w^2)^{j\beta_1} w^j.$$

Sia $j = \frac{\alpha N}{2} + j_1$. Allora, poichè $w^N = 1$ (e quindi $(w^N)^{\alpha\beta_1} = 1$)

$$(w^2)^{j\beta_1} = (w^2)^{\frac{\alpha N \beta_1}{2}}(w^2)^{j_1\beta_1} = (w^N)^{\alpha\beta_1}(w^2)^{j_1\beta_1} = (w^2)^{j_1\beta_1}.$$

Così se poniamo

$$\begin{aligned} \phi(j_1) &= \sum_{\beta_1=0}^{\frac{N}{2}-1} a_{2\beta_1}(w^2)^{j\beta_1},\ (j_1 = 0,1,\dots,\frac{N}{2}-1) && (7) \\ \psi(j_1) &= \sum_{\beta_1=0}^{\frac{N}{2}-1} a_{2\beta_1+1}(w^2)^{j\beta_1} && (8) \\ & && (9) \end{aligned}$$

abbiamo

$$c_j = \phi(j_1) + w^j \psi(j_1),\ j = 0,\dots,N-1.$$

Si noti che il calcolo di $\phi(j_1)$ e $\psi(j_1)$ significa che vengono compiute due analisi di Fourier con $\frac{N}{2} = 2^{k-1}$ termini invece di una con 2^k. Ora applichiamo la stessa idea alle due analisi di Fourier. Si hanno così 4 analisi di Fourier, ognuna ha 2^{k-2} termini; queste possono essere ulteriormente divise in 8 analisi di Fourier con 2^{k-3} termini, eccetera. Il numero di operazioni richieste per calcolare $\{c_j\}$ quando $\{\phi(j_1)\}$ e $\{\psi(j_1)\}$ sono stati calcolati, è al più $2 \cdot 2^k$ (un numero maggiore di calcoli può essere evitato se le potenze di w sono immagazzinate in memoria).
Sia p_k il numero totale di operazioni necessarie per calcolare i coefficienti quando $N = 2^k$. Come prima, abbiamo

$$p_k \le 2p_{k-1} + 2 \cdot 2^k,\ k = 1,2,\dots$$

Poichè $p_0 = 0$, si ha per induzione, essendo $N = 2^k$ e quindi $k = \log_2(N)$

$$\begin{aligned} p_k &\le 2p_{k-1} + 2\cdot 2^k \le 2\left(2p_{k-2} + 2\cdot 2^{k-1}\right) + 2\cdot 2^k \\ &\le 4\,p_{k-2} + 2\cdot 2\cdot 2^k = 4\left(2p_{k-3} + 2\cdot 2^{k-2}\right) + 4\cdot 2^k \\ &= 2^3 p_{k-3} + 8\cdot 2^{k-2} + 4\cdot 2^k = 2^3 p_{k-3} + 2\cdot 2^k + 2^2\cdot 2^k \\ &= 2^3 p_{k-3} + 2\cdot 3\cdot 2^k \le 2^k p_0 + 2k\cdot 2^k = 2N\cdot\log_2 N && (10) \end{aligned}$$

da paragonarsi con N^2 operazioni del metodo di base.

Per capire i vantaggi, facciamo qualche conto con Matlab, mostrando per alcune potenze di 2 (prima colonna), il numero di operazioni per il metodo di base (seconda colonna) e dell' FFT (terza colonna):

```
>> N=2.^(1:10);
>> N=N';
>> N2=N.^2;
>> Nlog=2*N.*log2(N);
>> [N N2 Nlog]

ans =

           2           4           4
           4          16          16
           8          64          48
          16         256         128
          32        1024         320
          64        4096         768
         128       16384        1792
         256       65536        4096
         512      262144        9216
        1024     1048576       20480

>>
```

3 FFT in Matlab

Vediamo come utilizzare in Matlab la FFT partendo dal relativo help

```
>> help fft

 FFT Discrete Fourier transform.
    FFT(X) is the discrete Fourier transform (DFT) of vector X.  For
    matrices, the FFT operation is applied to each column. For N-D
    arrays, the FFT operation operates on the first non-singleton
    dimension.

    FFT(X,N) is the N-point FFT, padded with zeros if X has less
    than N points and truncated if it has more.

    FFT(X,[],DIM) or FFT(X,N,DIM) applies the FFT operation across the
    dimension DIM.

    For length N input vector x, the DFT is a length N vector X,
    with elements
                     N
       X(k) =       sum  x(n)*exp(-j*2*pi*(k-1)*(n-1)/N), 1 <= k <= N.
                    n=1
```

```
    The inverse DFT (computed by IFFT) is given by
                     N
       x(n) = (1/N) sum  X(k)*exp( j*2*pi*(k-1)*(n-1)/N), 1 <= n <= N.
                    k=1

    See also IFFT, FFT2, IFFT2, FFTSHIFT.

 Overloaded methods
    help qfft/fft.m
```

Si capisce che tale procedura calcola, a partire da un vettore $x = (x(j))_{j=1,\ldots,N}$, il vettore $\mathscr{X} = (\mathscr{X}(k))_{k=1,\ldots,N}$ definito da

$$\mathscr{X}(\tau) = \sum_{n=1}^{N} x(n) \exp\left(\frac{-i\,2\pi\,(\tau-1)\,(n-1)}{N}\right), \tau = 1,\ldots,N. \tag{11}$$

(si noti che per errore nell'help di Matlab hanno indicato con j la costante immaginaria i).
Ci si domanda quale sia il legame di (11) con (5). Posto $n = \beta+1$, $j = \tau-1$, $x(n) = \frac{1}{N} f\left(\frac{2\pi(n-1)}{N}\right)$, abbiamo

$$c_{\tau-1} = \sum_{n=1}^{N} x(n) \exp\left(\frac{-i(\tau-1)2\pi(n-1)}{N}\right), \tau = 1,\ldots,N,$$

e quindi $\mathscr{X}(\tau) = c_{\tau-1}$.
Vediamo un esempio in Matlab, in cui analizziamo per $N = 8$ la funzione

$$f(x) = 5\exp(2\,i\,x)$$

in cui evidentemente $c_0 = c_1 = c_3 = \ldots = c_8 = 0$ e $c_2 = 5$. Quindi ci aspettiamo un vettore $\tau = \{\tau_k\}$ in cui poichè $\tau_k = c_{k-1}$ si ha che $\tau_3 = 5$ e tutte le altre componenti nulle.
Salviamo nel file `esempio_FFT`

```
clear all;
format long;

n=8;
f=inline('5*exp(i*2*t)');

h=2*pi/n;
t=(0:n-1)'*h;
ft=feval(f,t);

c_fft=fft(ft,n)*h/(2*pi);
```

e otteniamo come prevedibile

```
>> c_fft
c_fft =
 -0.00000000000000 + 0.00000000000000i
 -0.00000000000000 + 0.00000000000000i
  5.00000000000000 - 0.00000000000000i
  0.00000000000000 + 0.00000000000000i
  0.00000000000000 + 0.00000000000000i
  0.00000000000000 + 0.00000000000000i
                 0 + 0.00000000000000i
 -0.00000000000000 + 0.00000000000000i
```

Ora consideriamo

$$f(x) = 10\,\exp(-i\,2\,x)$$

e osserviamo che il risultato è

```
>> esempio_FFT
>> c_fft
c_fft =
 -0.00000000000000 - 0.00000000000000i
 -0.00000000000000 - 0.00000000000000i
                 0 - 0.00000000000000i
  0.00000000000000 - 0.00000000000000i
  0.00000000000000 - 0.00000000000000i
  0.00000000000000 - 0.00000000000000i
 10.00000000000000 + 0.00000000000000i
 -0.00000000000000 - 0.00000000000000i
```

A prima vista la cosa è sorprendente, visto che il coefficiente uguale a 10 appare alla settima componente, quella che dovrebbe essere di c_6 e non c_{-2}. Vediamo perchè . Si ha in generale per $f(x) = \lambda \exp(-i\,k\,x)$

$$\begin{aligned}
\mathscr{X}(\tau) &= \sum_{n=1}^{N} x(n)\,\exp\left(\frac{-i\,2\,\pi\,(\tau-1)\,(n-1)}{N}\right) \\
&= \frac{1}{N}\sum_{n=1}^{N} \lambda \exp\left(-i\,k\,\frac{2\pi(n-1)}{N}\right)\exp\left(\frac{-i\,2\,\pi\,(\tau-1)\,(n-1)}{N}\right) \\
&= \frac{1}{N}\sum_{n=1}^{N} \lambda \exp\left(-i\,2\,\frac{2\pi(\tau-1+k)(n-1)}{N}\right) \qquad (12)
\end{aligned}$$

sommatoria che vale λ se $\tau-1+k$ è multiplo di N (con τ avente valori tra 1 e N) e 0 altrimenti. Quindi nell'esempio precedente in cui $\lambda = 10$, $k = 2$, $N = 8$, la sommatoria vale 10 se $\tau - 1 + 2 = \tau + 1$ è multiplo di 8 (cioè $\tau = 7$) e 0 altrimenti.

Per convincersi di questo, posto $N = 8$ applichiamo la FFT a

$$f(x) = 10\,\exp(-6\,i\,x) + 20\,\exp(2\,i\,x).$$

Ci aspettiamo che il primo termine contribuisca alla sommatoria, cosicche' la terza componente valga 10 e il secondo termine contribuisca alla sommatoria, cosicche' la terza componente valga 20. Per linearità avremo quindi una terza componente uguale a 30, in quanto in generale

$$\sum_{n=1}^{N} (x(n) + y(n)) \exp(\ldots) = \sum_{n=1}^{N} x(n) \exp(\ldots) + \sum_{n=1}^{N} y(n) \exp(\ldots).$$

```
>> esempio_FFT
>> c_fft
c_fft =
 -0.00000000000000 - 0.00000000000000i
  0.00000000000000 - 0.00000000000000i
 30.00000000000000 + 0.00000000000000i
 -0.00000000000000 - 0.00000000000000i
  0.00000000000000 - 0.00000000000000i
 -0.00000000000000 + 0.00000000000000i
                 0 - 0.00000000000000i
  0.00000000000000 + 0.00000000000000i
>>
```

4 Online

Citiamo alcuni siti web che espongono parte dell'argomento:

http://en.wikipedia.org/wiki/Cooley-Tukey_FFT_algorithm
http://en.wikipedia.org/wiki/Fast_Fourier_transform
http://it.wikipedia.org/wiki/Rappresentazione_spettrale_dei_segnali

References

[1] K. Atkinson, *An Introduction to Numerical Analysis*, Wiley, (1989).

[2] J.W. Cooley e J.W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Math. Comput. 19, p. 297-301, (1965).

[3] G. Dahlquist e A. Bjorck , *Numerical methods*, Dover, (2003).

[4] S.D. Conte e C. de Boor, *Elementary Numerical Analysis, An Algorithmic Approach*, McGraw-Hill, (1981).

[5] G. Gilardi *Analisi Due, seconda edizione*, McGraw-Hill, (1996).

[6] D.H. Griffel, *Applied functional analysis*, Dover publications, 2002.

[7] A.N. Kolmogorov e S.V. Fomin, *Introductory Real Analysis*, Dover publications, 1970.

[8] A. Quarteroni, R. Sacco e F. Saleri *Matematica Numerica*, Springer, (1998).

[9] M.R. Spiegel *Variabili Complesse*, McGraw-Hill, collana Schaum, (1994).