Il restauro di immagini digitali

Dario A. Bini, Università di Pisa

4 dicembre 2013

1 Introduzione

In campo fotografico e cinematografico la tecnologia digitale ha da tempo soppiantato la vecchia tecnologia analogica. Macchine fotografiche digitali vengono comunemente e diffusamente utilizzate a livello amatoriale e professionale.

Una foto digitale in bianco/nero, nel formato raw, consiste di una matrice $m \times n$ di numeri interi $F = (f_{i,j})$ in cui il numero $f_{i,j}$ descrive la luminosità del punto dell'immagine in posizione (i, j). Generalmente si fa corrispondere il valore 0 al nero e il valore massimo consentito al bianco. Il valore massimo consentito è un intero minore di 65536, generalmente si usa 255.

Una foto a colori, nella sua rappresentazione RGB viene registrata come una terna di matrici R, G, B di dimensione $m \times n$, ciascuna delle quali esprime la luminosità rispettivamente del canale del rosso (Red), verde (Green) e blu (Blue). Infatti il generico punto in posizione (i, j) di una immagine digitale a colori è visto come la somma di una componente di rosso $r_{i,j}$, una di verde $g_{i,j}$ e una di blu $b_{i,j}$. Anche in questo caso si conviene che il valore 0 corrisponde a intensità di colore nulla, e il valore massimo corrisponde all'intensità massima.

I formati jpg o jpeg in cui si trovano registrate le foto comunemente fornite dalle macchine digitali sul mercato sono dei formati compressi della versione originale in formato raw.

La disponibilità dell'informazione fotografica data in termini numerici permette di effettuare in modo molto agevole elaborazioni dell'immagine che una volta erano solo consentite in camera oscura. Ad esempio, è possibile schiarire o scurire parte dell'immagine, oppure alterare selettivamente il suo contrasto, cambiare la saturazione o l'equilibrio cromatico, ottenere effetti di solarizzazione.

Nel seguito ci riferiremo ai singoli punti dell'immagine, i cui valori numerici determinano l'intensità luminosa e la tonalità del colore, come ai *pixel*. Il termine pixel è la contrazione dell'espressione inglese *picture-element*, elemento dell'immagine. Ci riferiremo al *supporto dell'immagine* quando si vuole indicare l'insieme dei valori degli indici dei pixel dell'immagine.

Utilizzeremo inoltre lo standard PGM per codificare immagini in bianco/nero e lo standard PPM per le immagini a colori. Per dettagli su questi standard si rimanda al sito `http://netpbm.sourceforge.net`.

Nel formato PGM (Portable Gray Map), una immagine di $m \times n$ pixel in bianco e nero viene memorizzata come una matrice $m \times n$ di numeri interi.

Una immagine a colori viene memorizzata nel formato PPM (Portable Pix Map) con tre matrici di interi, la prima registra le intensità del rosso la seconda del verde e la terza del blu. Il valore zero corrisponde alla minima intensità di colore mentre il valore massimo consentito, un intero minore di 65536 (generalmente 255), corrisponde alla massima intensità di colore. La modalità di rappresentazione di immagini a colori mediante le componenti di rosso, verde e blu è detta RGB dal nome dei tre colori Red, Green, Blue.

I file PGM e PPM, e in generale la classe di tutti i file di questo tipo denotata con l'acronimo PNM Portable aNy Map, hanno due modalità di uso: la modalità ASCII e la modalità RAW. Nella seconda i valori dei pixel, ad esempio 123, vengono memorizzati con un solo byte per cui il file ha l'ingombro di memoria pari al numero totale dei pixel. Nella modalità ASCII invece ciascun pixel, ad esempio 123, viene memorizzato scrivendo ciascuna cifra in base 10, nel caso di 123 i tre numeri 1,2,3 seguiti da un carattere separatore (spazio o ritorno a capo). Per cui, essendo ciascun carattere alfanumerico memorizzato con un byte, l'ingombro di un file nella modalità ASCII è al più 4 volte maggiore. Si osserva infatti che al più 4 byte sono sufficienti per rappresentare il valore di un pixel in base 10.

Diamo una breve spiegazione di come sono organizati i file PNM. Per maggiori dettagli si rimanda a netpbm.sourceforge.net.

Un file PNM deve contenere

- Un "magic number" per identificare il tipo di file (vedi tabella riportata di seguito). Ad esempio per un file pgm il magic number è la stringa "P2".
- Spazi vuoti (blanks, TABs, CRs, LFs).
- La larghezza in pixel dell'immagine, in caratteri ASCII, in decimale.
- Spazio vuoto.
- L'altezza in pixel dell'immagine, sempre in ASCII decimale.
- Spazio vuoto.
- Il massimo valore di grigio in ASCII decimale. Deve essere inferiore a 65536, e maggiore di zero (generalmente 255).
- Un solo Spazio vuoto, generalmente un ritorno a capo.
- Un insieme di righe ciascuna consistente in un insieme di valori numerici compresi tra 0 e il massimo valore di grigio assegnato (un intero minore di 65536), dove 0 corrisponde al nero e il massimo valore assegnato corrisponde al bianco. Tali valori descrivono i pixel dell'immagine a partire dal pixel in alto a sinistra, procedendo per righe. Nella modalità ASCII i valori numerici sono rappresentati in ASCII decimale e ciascun valore è separato da uno spazio vuoto. In modalità RAW, se il massimo valore è

P2	file PGM, modalità ASCII
P5	file PGM, modalità RAW
P3	file PPM, modalità ASCII
P6	file PPM, modalità RAW

Tabella 1: Lista dei "numeri magici"

inferiore a 256 ciascun valore è rappresentato con un byte. Se il valore massimo è maggiore di 255 allora ciascun valore è rappresentato da due byte. Il più significativo è il primo.

La tavola dei numeri magici è riportata nella tabella 1.

2 Il modello di sfocatura

Una delle applicazioni più interessanti della tecnologia digitale è il restauro di immagini alterate da difetti di messa a fuoco o da difetti dovuti a movimenti della macchina fotografica all'atto dello scatto effettuato con tempi di otturazione troppo lunghi.

Fotografie che sono venute sfocate per un diffetto di messa a fuoco dell'obiettivo o per movimenti della fotocamera possono essere in qualche modo rimesse a fuoco se si conosce il tipo di sfocatura. Supponiamo di avere scattato una foto ad una immagine costituita da un solo punto luminoso di intensità unitaria come in figura 1. Con l'obiettivo regolato male si otterrà qualcosa tipo l'immagine in figura 2 dove abbiamo volutamente esagerato l'effetto della sfocatura. Questa nuova immagine sarà definita da dei pixel $f_{i,j}$, dove per comodità facciamo scorrere gli indici i e j da $-k$ a k dove $2k+1$ è l'ampiezza del supporto dell'immagine del puntolino sfocato. La tabella di numeri $f_{i,j}$ per $i, j = -k, k$ descrive quindi il tipo di sfocatura ed è chiamata in gergo *Point-Spread Function*, o più semplicemente PSF. Essa descrive il modo in cui il punto luminoso si spande in una macchiolina. I valori $f_{i,j}$ sono non negativi e devono sommarsi ad 1 poiché la quantità di luminosità presente nell'immagine sfocata deve essere uguale a quella presente nell'immagine originale.

Allora, se $A = a_{i,j}$ per $i, j \in \mathbb{Z}$, sono i pixel di una immagine originale A costituita da infiniti pixel collocati su un piano, possiamo identificare una fotografia di A come una matrice $B = (b_{i,j})$ che cattura una porzione rettangolare di A, precisamente quella i cui indici sono $i = 1, \ldots, m$, $j = 1, \ldots, n$, cioè $b_{i,j} = a_{i,j}$ per $i = 1, \ldots, m$, $j = 1, \ldots, n$. Ora supponiamo che tutti i pixel di A siano nulli tranne $a_{p,q} = g \neq 0$, cioè l'immagine è costituita da un unico punto luminoso di intensità g posto nella posizione (p, q). Se la foto viene scattata con l'obiettivo fuori fuoco l'immagine sfocata $b_{i,j}$ avrà pixel uguali a

$$b_{i,j} = \begin{cases} g f_{i-p,j-q}, & \text{se } |i-p|, |j-q| \leq k \\ b_{i,j} = 0, & \text{altrove.} \end{cases}$$

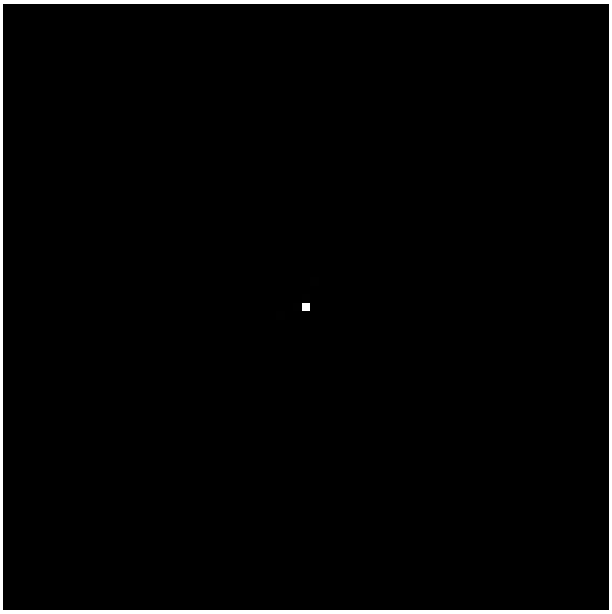

Figura 1: Immagine di un punto luminoso

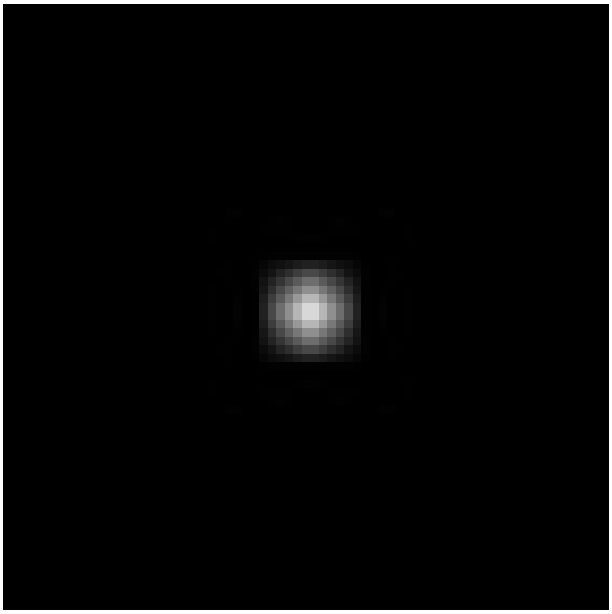

Figura 2: Immagine di un punto luminoso fotografato con l'obiettivo fuori fuoco

In questo modello il tipo di sfocatura (di PSF) è invariante spazialmente, cioè non dipende dalla posizione del punto (p, q) ed è definito dalla stessa tabella di numeri $(f_{i,j})_{i,j=-k,k}$qualunque sia il punto (p, q). Il modello fornisce una approssimazione della situazione reale. Infatti, nella realtà il comportamento di un obiettivo fotografico è diverso a seconda che si consideri il centro o il bordo dell'immagine.

Un'altra considerazione riguarda il supporto dell'immagine che per A è dato da $Z\times\mathbb{Z}$. Però a incidere sui pixel dell'immagine sfocata intervengono solo i pixel $a_{i,j}$ di A tali che $i = -k+1, \ldots, m+k$, $j = -k+1, \ldots, n+k$. Cioè l'immagine sfocata viene originata non solo dai pixel che hanno indici $1 \le p \le m$, $1 \le j \le n$, ma anche dai pixel che sono fuori da questo supporto purché rimangano dentro una cornice di ampiezza k. Infatti, se il pixel luminoso di coordinate (p, q) si trova a distanza al più k (semiampiezza del supporto della PSF) dal supporto $[1 : m] \times [1 : n]$, allora l'effetto della sfocatura si ritrova anche dentro il supporto di B.

In questo modello l'immagine sfocata di una immagine generica costituita da più pixel non nulli sarà la somma delle immagini che si ottengono sfocando i singoli pixel. Vale cioè

$$b_{i,j} = \sum_{p,q} f_{i-p,j-q} a_{p,q}, \quad i = 1, \ldots, m,\ j = 1, \ldots, n, \tag{1}$$

dove la somma è fatta per $p = 1-k, \ldots, m+k$, $q = 1-k, \ldots, n-k$, su tutti gli indici per cui $i-p$ e $j-q$ sono compresi tra $-k$ e k. Ponendo $r = i-p$ e $s = j-q$, la (1) può essere riscritta come

$$b_{i,j} = \sum_{r,s=-k}^{k} f_{r,s} a_{i-r,j-s}, \quad i = 1, \ldots, m,\ j = 1, \ldots, n. \tag{2}$$

Si riportano alcuni esempi di PSF.

- PSF uniforme: in questo caso la matrice F di dimensioni $(2k+1)\times(2k+1)$ ha tutti elementi uguali tra di loro che valgono quindi $f_{i,j} = 1/(2k+1)^2$. Un punto viene quindi trasformato in un rettangolo di $(2k+1) \times (2k+1)$ pixel. Maggiore è k e maggiore è l'effetto di sfocatura.

- PSF esponenziale: i valori della PSF sono dati da $f_{i,j} = \theta e^{-\alpha(i^2+j^2)}$, per $i, j = -k, \ldots, k$, dove θ è un parametro di normalizzazione scelto in modo che la somma degli $f_{i,j}$ faccia 1, e α è un numero positivo che determina l'ampiezza dell'effetto di sfocatura. In questo caso il punto centrale della PSF, quello corrispondente agli indici $i = j = 0$, è quello di luminosità massima, inoltre la luminosità decade esponenzialmente a zero man mano che ci si allontana dal punto centrale. Il decadimento dipende dalla grandezza di α. Per $\alpha = 0$ si ottiene la PSF uniforme. La figura 3 mostra il caso in cui $k = 10$ e $\alpha = 0.05$.

- PSF sinc: i valori della PSF di tipo sinc sono dati da $f_{i,j} = \theta(\sigma+\mathrm{sinc}(\alpha(i^2+j^2)^{1/2}))$ dove la funzione sinc è data da $\mathrm{sinc}(x) = \sin x/x$. Questo tipo di

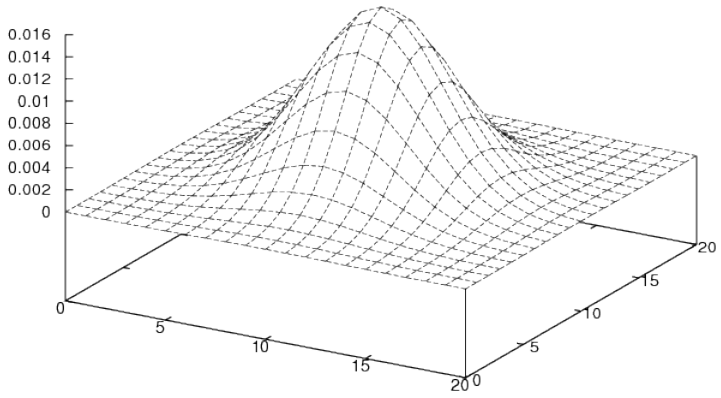

Figura 3: Point-Spread Function di tipo esponenziale

PSF è quello più vicino alla situazione reale delle macchine fotografiche digitali. Il parametro α determina l'intensità della sfocatura: più piccolo è α e maggiore è la sfocatura. Il parametro σ è scelto in modo da avere valori non negativi della PSF, il valore θ viene scelto in modo che gli elementi della PSF si sommino a 1. Si riporta nella figura 4 il grafico della funzione $\mathrm{sinc}(2x)$.

Nella figura 5 si riporta il grafico della PSF bidimensionale di tipo sinc con $\alpha = 1$.

Dal punto di vista computazionale per sfocare una immagine in bianco e nero basta calcolare una doppia sommatoria: data $A = (a_{i,j})$ e $F = (f_{i,j})$, si calcola $B = (b_{i,j})$ mediante la (1) o la (2). Per rimettere a fuoco una immagine basta "risolvere" un sistema lineare di mn equazioni in $(m+2k)(n+2k)$ incognite: data l'immagine sfocata B e la PSF F si calcola l'immagine originale A risolvendo il sistema (2).

Per immagini a colori la manipolazione va compiuta sui tre canali del rosso, verde e blu. Per una foto scattata con un macchina fotografica da 5 megapixel si devono risolvere tre sistemi di circa 5 milioni di equazioni e di incognite per poter rimettere a fuoco la foto.

I sistemi che si ottengono in questo modo sono sottodeterminati, hanno cioè meno equazioni che incognite. La risoluzione di questi sistemi va allora intesa in termini di "risoluzione ai minimi quadrati". Il sistema (2) può essere scritto in forma matriciale come $Ha = b$ dove a e b sono i vettori ottenuti giustapponendo

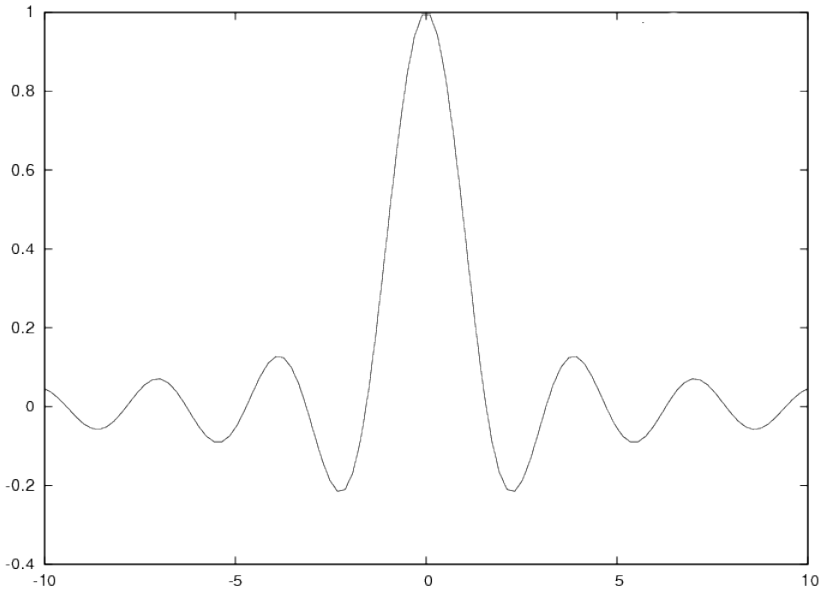

Figura 4: Point-Spread Function di tipo sinc: $\sin(2x)/(2x)$

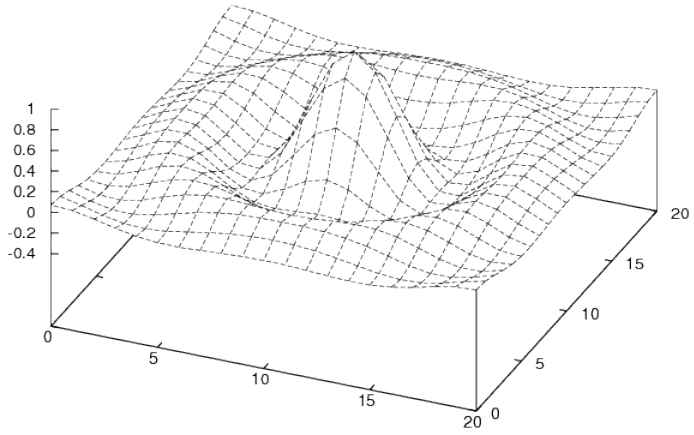

Figura 5: Point-Spread Function di tipo sinc: $\sin((x^2+y^2)^{\frac{1}{2}})/((x^2+y^2)^{\frac{1}{2}})$

una sull'altra rispettivamente le colonne di A e di B. Mentre H è la matrice del sistema lineare che si ottiene con questa organizzazione in vettori. Più avanti analizziamo la struttura di H.

Si osserva che, usando un linguaggio di programmazione di alto livello quale il Fortran 90, il procedimento di sfocatura si realizza in modo abbastanza semplice. La seguente subroutine realizza giusto questo.

```
! Sfoca l'immagine a e la mette in b usando la psf assegnata
! Calcola cioe' il prodotto matrice vettore b=Ha
subroutine ax(m,n,k,psf,a,b)
  implicit none
  integer, parameter :: dp=kind(0.d0)
  integer :: m,n,k
  real(dp),dimension(-k:k,-k:k)       :: psf
  real(dp),dimension(1:m,1:n)         :: b
  real(dp),dimension(1-k:m+k,1-k:n+k) :: a
  integer :: i,j,p,q
  b=0
  do i=1,m
     do j=1,n
        do p=-k,k
           do q=-k,k
              b(i,j)=b(i,j)+psf(p,q)*a(i-p,j-q)
           end do
        end do
     end do
  end do
end subroutine ax
```

Il programma seguente costruisce una PSF, la salva nel file `psf.txt`, legge una immagine dal file `foto.pgm`, la sfoca e la salva nel formato PGM nel file `sfocata.pgm`.

```
program leggi_e_sfoca
  implicit none
  integer, parameter :: dp=kind(0.d0)
  integer :: m, n, i, j, k, liv
  real(dp), allocatable :: a(:,:), b(:,:), psf(:,:)
  integer :: aux
  character(len=2) :: ch
! costruisce la psf (2k+1)x(2k+1) con valori identici
  k=4
  allocate(psf(-k:k,-k:k))
  psf=1.d0
  psf=psf/sum(psf)
! salva la pfs
  open(unit=10, file='psf.txt', action='write')
```

```
  write(10,*)k
  do i=-k,k
    do j=-k,k
      write(10,*) psf(i,j)
    end do
  end do
! apre il file e legge l'immagine
  open(unit=20, file='foto.pgm', action='read')
  read(20, *) ch
  read(20,*) n1, m1
  read(20,*) liv
  n=n1-2*k
  m=m1-2*k
  allocate(a(1-k:m+k,1-k:n+k))
  do i=1-k,m+k
    do j=1-k,n+k
      read(20,*) a(i,j)
    enddo
  enddo
! sfoca l'immagine
  allocate(b(1:m,1:n))
  call ax(m,n,k,a,psf,b)
! scrive l'immagine sfocata
  open(unit=30, file='sfocata.pgm', action='write')
  write(30,'(A)') ch
  write(30,*) n,m
  write(30,*) liv
  do i=1,m
    do j=1,n
      aux=b(i,j)
      aux=max(0,aux)
      aux=min(255,aux)
      write(30,*)aux
    enddo
  enddo
end program leggi_e_sfoca
```

Il formato '(A)' usato nella scrittura del "numero magico" P2 contenuto nella variabile ch, serve a evitare che venga inserito uno spazio davanti alla lettera P. Ciò accadrebbe se usassimo il formato libero di scrittura con l'istruzione `write(30,*)`. La presenza di questo spazio non desiderato impedirebbe al visualizzatore del file, ad esempio `display` di riconoscere che si tratta di un file pgm. Un'altra osservazione di carattere tecnico è che molti editori e manipolatori di file contenenti immagini nel creare un file pgm o pnm inseriscono dentro il file dei commenti costituiti da righe precedute dal carattere #. Il programma scritto non prevede la presenza di tali commenti che andranno quindi rimossi

Figura 6: Fotografia originale

prima di lanciare il programma.

Ecco come appare l'immagine che si ottiene sfocando con una PSF uniforme di semi ampiezza $k = 8$ la seguente fotografia scattata ad una pianta di peperoni habanero. La fotografia in origine era di 2288×1712 pixel, scattata da una macchina fotografica da 4 megapixel, ed è qui stata ridotta a 800×600 pixels.

3 Matrice del sistema

Vediamo di descrivere la struttura della matrice H del sistema $Ha = b$ che lega l'immagine originale A e l'immagine sfocata B organizzate come vettori rispettivamente a e b.

Per far questo esaminiamo prima il caso più elementare in cui $n = 1$, cioè le immagini A, B e la PSF F sono costituite da una sola colonna, cioè $A = (a_{k-1}, \ldots, a_{m+k})^T$, $B = (b_1, \ldots, b_m)^T$, $F = (f_{-k}, \ldots, f_k)^T$. In questo caso il sistema (2) prende la forma

$$b_i = \sum_{r=-k}^{k} f_r a_{i-r}, \quad i = 1, \ldots, m.$$

Si può osservare che in ciascuna equazione di questo sistema compaiono $2k+1$ incognite consecutive, ad esempio nella i-esima equazione sono $a_{i-k}, \ldots, a_{i+k}$ e che i coefficienti di queste incognite in ciascuna equazione sono sempre gli stessi, cioè gli elementi della PSF. Questo fatto comporta che la matrice del sistema ha

Figura 7: Fotografia sfocata

una struttura a banda di ampiezza $2k+1$ e che gli elementi della matrice sulla banda sono uguali lungo le diagonali. Cioè, in notazione matriciale, il sistema ha la forma

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} f_k & f_{k-1} & \cdots & f_{-k+1} & f_{-k} & & & \\ & f_k & f_{k-1} & \ddots & f_{-k+1} & f_{-k} & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & f_k & f_{k-1} & \cdots & f_{-k+1} & f_{-k} \end{bmatrix} \begin{bmatrix} a_{1-k} \\ \vdots \\ a_1 \\ a_2 \\ \vdots \\ a_m \\ \vdots \\ a_{m+k} \end{bmatrix}$$

Matrici i cui elementi sono uguali lungo le diagonali, o più precisamente, i cui elementi dipendono unicamente dalla differenza degli indici di riga e colonna vengono dette *matrici di Toeplitz* Queste matrici hanno speciali proprietà computazionali e spettrali.

Nel caso generale in cui $m, n > 1$, se scriviamo la matrici A e B rispettivamente come vettori a e b ottenuti incolonnando una sull'altra le colonne rispettivamente di A e di B, la matrice del sistema H prende la forma di matrice $(n+2k) \times n$ a blocchi

$$\begin{bmatrix} F_k & F_{k-1} & \dots & F_{-k+1} & F_{-k} & & & \\ & F_k & F_{k-1} & \ddots & F_{-k+1} & F_{-k} & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & F_k & F_{k-1} & \dots & F_{-k+1} & F_{-k} \end{bmatrix}$$

i cui blocchi F_r sono matrici $(m+2k) \times m$ dati da

$$F_r = \begin{bmatrix} f_{r,k} & f_{r,k-1} & \dots & f_{r,-k+1} & f_{r,-k} & & & \\ & f_{r,k} & f_{r,k-1} & \ddots & f_{r,-k+1} & f_{r,-k} & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & f_{r,k} & f_{r,k-1} & \dots & f_{r,-k+1} & f_{r,-k} \end{bmatrix}$$

Cioè la matrice del sistema ha due livelli di struttura: in termini di blocchi e in termini di elementi dei blocchi. Infatti, la matrice è di Toeplitz a blocchi con blocchi di Toeplitz; è inoltre a banda a blocchi con blocchi a banda.

Si osserva che la risoluzione ai minimi quadrati del sistema $Ha = b$ associato alla rimessa a fuoco dell'immagine sfocata in linea teorica potrebbe essere affrontato mediante l'uso della SVD. Però le dimensioni della matrice H del sistema che sono $(m+2k)m \times (n+2k)n$ diventano proibitive per l'alto costo computazionale. Ad esempio, per una immagine piccola di 1024×768 pixel, con una PSF 15×15 la matrice del sistema ha dimensione 612864×1079296 e, contando che il numero di operazioni per calcolare la SVD cresce col cubo delle dimensioni, il costo dell calcolo della SVD sarebbe dell'ordine di grandezza di 10^{18}. Un computer che esegue 10^9 operazioni al secondo richiederebbe 10^9 secondi, cioè più di 3 anni di tempo di CPU.

Un approccio alternativo consiste nel procedere nel seguente modo. La soluzione ai minimi quadrati del sistema $Ha = b$ si può scrivere come

$$a = H^+ b$$

dove

$$H^+ = H^T (HH^T)^{-1}$$

è l'inversa generalizzata della matrice H del sistema. Per cui possiamo prima calcolare la soluzione del sistema

$$HH^T x = b \tag{3}$$

e poi calcolare il prodotto $a = H^T x$.

Il sistema (3), che ha matrice definita positiva, si risolve applicando un metodo iterativo.

Come metodo iterativo può essere applicato il metodo di Richardson

$$a^{(\nu+1)} = a^{(\nu)} - \theta(HH^T a^{(\nu)} - b)$$

dove θ è un parametro che rende la matrice di iterazione $P = I - \theta HH^T$ di raggio spettrale minore di 1, ad esempio $\theta = 1/\|HH^T\|$ per una qualsiasi norma indotta $\|\cdot\|$. Il metodo di Richardson in ambiente di "image restoration" è più noto vome "metodo di Landweber". Una alternativa più efficace ìl metodo del gradiente coniugato.

Metodi di questo tipo hanno come operazione più costosa, il calcolo del prodotto tra la matrice HH^T ed un vettore. Questo prodotto, $z = HH^T x$ può essere calcolato valutando separatamente i prodotti $y = H^T x$ e $z = Hy$. Quest'ultimi due prodotti possono essere calcolati a basso costo sfruttando la particolare struttura della matrice H.

Ad esempio, il calcolo del prodotto Hz è realizzato dalla subroutine `ax` di sfocatura che abbiamo descritto e comporta il calcolo di $(2k+1)^2$ moltiplicazioni e di circa altrettante addizioni per pixel per un totale di $2mn(2k+1)^2$ operazioni aritmetiche. Se il valore di k fosse troppo grande per cui il costo diventa insostenibile, si può utilizzare un algoritmo basato sulla FFT che calcola questo prodotto con circa $\frac{3}{2}\hat{n}\hat{m}\log_2(\hat{m}\hat{n})$ operazioni aritmetiche dove $\hat{n}$ e $\hat{m}$ sono le più piccole potenze intere di 2 maggiori o uguali rispettivamente a m e n. Vedremo questo algoritmo nella prossima sezione.

Per implementare il metodo del gradiente coniugato così come il metodo di Richardson abbiamo bisogno anche di una subroutine che moltiplichi la matrice H^T per un vettore. Vediamo ora come si può realizzare ciò. Esaminiamo prima il caso di $n = 1$ per cui se $y = H^T x$ allora

$$\begin{bmatrix} y_{1-k} \\ \vdots \\ y_{m+k} \end{bmatrix} = \begin{bmatrix} f_k & & & & & & \\ \vdots & \ddots & & & & & \\ f_{-k} & \cdots & f_k & & & & \\ & f_{-k} & f & \ddots & f_k & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & \ddots & \ddots & f_k \\ & & & & & \ddots & \vdots \\ & & & & & & f_{-k} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

per cui la formula diventa

$$y_i = \sum_{r=-k}^{k} f_{-r} x_{i-r}, \qquad i = 1-k, \ldots, m+k$$

dove si assume $x_r = 0$ se $r \leq -k$ o se $r > m + k$.

Nel caso bidimensionale si ha

$$y_{i,j} = \sum_{r,s=-k}^{k} f_{r,-s} x_{i-r,j-s}$$

dove si assume $x_{r,s} = 0$ se $r \leq -k$ o $r > m + k$, o $s \leq -k$ o $s > n + k$.

Si osservi che la formula è analoga alla (2) dove gli indici di f vengono cambiati di segno.

Una subroutine in Fortran 90 che implementa questa formula è la seguente

```
! Calcola il prodotto y=H^Tx
subroutine atx(m,n,k,psf,x,y)
  implicit none
  integer, parameter :: dp=kind(0.d0)
  integer :: m,n,k
  real(dp),dimension(-k:k,-k:k)       :: psf
  real(dp),dimension(1:m,1:n)         :: x
  real(dp),dimension(1-k:m+k,1-k:n+k) :: y
  integer :: i,j,p,q
  y=0
  do i=1-k,m+k
     do j=1-k,n+k
        do p=-k,k
           do q=-k,k
              if(i-p>0 .and. i-p<=m .and. j-q>0 .and. j-q<=n)then
                 y(i,j)=y(i,j)+psf(-p,-q)*x(i-p,j-q)
              endif
           end do
        end do
     end do
  end do
end subroutine atx
```

A questo punto è immediato scrivere una subroutine che esegue un numero assegnato `iter` di iterazioni del metodo di Richardson

```
! subroutine che risolve il sistema H^THx=b col metodo
! di Richardson
subroutine richardson(m,n,k,iter,psf,b,a)
  ! esegue iter passi dell'iterazione di Richardson
  ! con theta uguale a 1
  implicit none
  integer,parameter :: dp=kind(0.d0)
  integer           :: m,n,k,iter
  real(dp)          :: psf(-k:k,-k:k), a(1-k:m+k,1-k:n+k), b(m,n)
  ! variabili locali
  integer           :: it
  real(dp)          :: theta, aux(1-k:m+k,1-k:n+k), aux1(m,n)

 write(*,*)"assegna il valore del parametro theta in (0,2)"
  read(*,*)theta
  theta=theta/sum(psf)
  theta=1.d0/sum(psf)
```

Figura 8: Richardson 20

```
  a=0.d0
  do it=1,iter
    call ax(m,n,k,a,psf,aux1)
    aux1=aux1-b
    call atx(m,n,k,aux,psf,aux)
    a=a-aux*theta
  end do
end subroutine richardson
```

Un programma che legge un'immagine, la sfoca e poi invoca il metodo di Richardson per approssimare la soluzione ai minimi quadrati è riportato di seguito.

```
programma
```

Riportiamo qui sotto le immagini rimesse a fuoco usando il metodo di Richardson con un numero di iterazioni pari a 20, 40, 80, 160.

Si può notare che nelle immagini che approssimano la soluzione, fornite dal metodo iterativo di Richardson, il valore approssimato della soluzione presenta degli "echi" in prossimità delle linee in cui si trova una brusca variazione di luminosità. Questo fenomeno è riconducibile all'effetto Gibbs che produce i cosiddetti Ringing Artifacts. La motivazione di questo fenomeno in termini dell'effetto Gibbs sarà chiara nella sezione sulla regolarizzazione.

Figura 9: Richardson 40

Figura 10: Richardson 80

Figura 11: Richardson 160

4 Il metodo del gradiente coniugato

Il metodo di Richardson che abbiamo usato nella sezione precedente è un metodo stazionario che ha una convergenza regolata dal raggio spettrale della matrice di iterazione. Infatti il raggio spettrale esprime la riduzione asintotica media dell'errore di approssimazione per passo. Più piccolo è il raggio spettrale e più alta è la velocità di convergenza. Ad esempio, un raggio spettrale pari a 0.999 richiede circa 700 iterazioni per dimezzare l'errore iniziale.

Un metodo dotato di proprietà di convergenza più interessanti è il metodo del gradiente coniugato. Il metodo di fatto è un metodo diretto, infatti se applicato a un sistema con matrice definita positiva di dimensioni $n \times n$ impiega n passi per fornire la soluzione in modo esatto, naturalmente con una aritmetica non affetta da errori. Però l'errore di approssimazione decresce velocemente a zero col numero di passi effettuati per cui, anche dopo un numero relativamente basso di passi si ottengono approssimazioni molto buone della soluzione. Per le proprietà del metodo del gradiente coniugato si rimanda alle numerose pubblicazioni che esistono al riguardo. In questo articolo ci limitiamo a descrivere il metodo dal punto di vista algoritmico utilizzando la formulazione data in Templates for the solution of linear systems: building blocks for iterative methods.

1. Si fissa $\epsilon > 0$ e si sceglie $x^{(0)}$ a caso
2. $r^{(0)} = bAx^{(0)}$
3. **for** $i = 1, 2, \ldots$

4. $z^{(i1)} = r^{(i1)}$
5. $\rho_{i1} = r^{(i1)T} z^{(i1)}$
6. **if** $i = 1$
7. $p^{(1)} = z^{(0)}$
8. **else**
9. $\beta_{i1} = \rho_{i1}/\rho_{i2}$
10. $p^{(i)} = z^{(i1)} + \beta_{i1} p^{(i1)}$
11. **end if**
12. $q^{(i)} = Ap^{(i)}$
13. $\alpha_i = \rho_{i1}/p^{(i)T} q^{(i)}$
14. $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$
15. $r^{(i)} = r^{(i1)} \alpha_i q^{(i)}$
16. **if** $\|r^{(i)}\| \leq \epsilon$ arresta le iterazioni
17. **end if**
18. **end for**

Si riportano i risultati ottenuti col metodo del gradiente coniugato applicato con 20,40, 80 e 160 iterazioni.

Si può notare come le approssimazioni fornite dal gradiente coniugato siano migliori, a parità di passi, di quelle fornite dal metodo di Richardson.

5 Il rumore e la regolarizzazione

Nella risoluzione di un sistema del tipo $Hx = b$ proveniente dalle applicazioni il termine noto b è generalmente affetto da errore. Questo errore può essere causato dal sistema fisico di rivelazione, tipo il sensore di una macchina fotografica, o può essere causato dal "rumore di quantizzazione" numerica dovuto al fatto che l'intensità luminosa viene rappresentata in modo discreto con un intero compreso tra 0 e 255. Questo comporta che il sistema che dovremmo in effetti risolvere è del tipo $Hy = b - \tau$ dove b è il dato di cui disponiamo ma che è affetto da un errore τ che non si conosce per cui $b - \tau$ è la vera immagine sfocata.

Di fatto andiamo a calcolare la soluzione dei minimi quadrati $x = H^+ b$ dove $H^+ = H^T (HH^T)^{-1}$, mentre avremmo voluto calcolare $y = H^+ (v - \tau)$.

Un semplice conto mostra che la differenza tra la soluzione x che avremmo voluto calcolare e quella y che possiamo calcolare è data da

$$y - x = H^+ \tau$$

Figura 12: Gradiente coniugato dopo 20 iterazioni senza rumore

Figura 13: Gradiente coniugato dopo 40 iterazioni senza rumore

Figura 14: Gradiente coniugato dopo 80 iterazioni senza rumore

Figura 15: Gradiente coniugato dopo 160 iterazioni senza rumore

Se l'errore τ è piccolo in norma ma la matrice H è mal condizionata allora questo errore può essere molto grande in norma, tanto più grande quanto maggiore è il numero di condizionamento di H. A questo riguardo possiamo fare la seguente analisi più precisa. Se $H = U\Sigma V^T$ è la SVD di H allora

$$H = \sum_i \sigma_i u_i v_i^T, \quad H^+ = \sum_i \sigma_i^{-1} u_i v_i^T,$$

dove u_i e v_i sono le colonne rispettivamente di U e V. Per cui si ottiene la seguente rappresentazione dell'errore in termini dei valori e dei vettori singolari di H

$$y - x = \sum_i \sigma_i^{-1} (v_i^T \tau) u_i.$$

Si scopre allora che l'errore τ viene maggiormente amplificato lungo le componenti associate a quei vettori singolari di H che corrispondono ai valori singolari più piccoli. Infatti, la componente dell'errore $(v_i^T \tau)u_i$ lungo il vettore singolare u_i viene amplificata dal fattore σ_i^{-1}. Tanto più σ_i è piccolo e tanto più grande è l'amplificazione dell'errore.

Generalmente per le matrici generate da PSF che sfocano, e quindi attenuano le alte frequenze, i vettori singolari che corrispondono ai valori singolari più piccoli sono vettori di alta frequenza, cioè le loro componenti hanno un numero alto di variazioni e contribuiscono a creare il micro dettaglio dell'immagine. Accade quindi che la parte del rumore si nota come una granulosità fitta e sottile.

Per sistemi con matrice simmetrica e definita positiva, come ad esempio la matrice HH^T, i vettori singolari coincidono con gli autovettori della matrice del sistema e i valori singolari con gli autovalori. Infati dalla decomposizione spettrale della matrice H si ha $HH^T = UDU^T$ con D la matrice diagonale con gli autovalori di H sulla diagonale principale e U matrice ortogonale. Più precisamente U è la matrice dei vettori singolari di H e $D = \text{diag}(\sigma_1^2, \ldots, \sigma_{mn}^2)$.

Alcuni metodi iterativi, come il metodo di Richardson o il metodo del gradiente coniugato, hanno la caratteristica per cui le componenti dell'errore di approssimazione lungo gli autovettori corrispondenti agli autovalori grandi converge molto rapidamente a zero mentre le componenti dell'errore lungo gli autovettori corrispondenti agli autovalori piccoli convergono a zero molto lentamente. Questo fa si che nelle prime iterazioni si approssimano bene le componenti in bassa frequenza dell'immagine mentre le componenti di frequenza elevata richiedono molte iterazioni per essere approssimate. Poiché il rumore è annidato nelle componenti in alta frequenza, usando questi metodi ed evitando di eseguire un numero troppo elevato di iterazioni si riesce a mettere ben a fuoco l'immagine senza però amplificare troppo il rumore.

Questo fenomeno è mostrato nelle seguenti figure dove si riportano le immagini ottenute con iterazioni del gradiente coniugato a partire dall'immagine sfocata in cui i valori numerici sono stati approssimati alla parte intera e quindi sono affetti da un errore τ.

Cioè si applica il seguente programma che legge prima l'immagine sfocata e troncata alla parte intera e poi risolve il problema dei minimi quadrati di minima norma usando il metodo del gradiente coniugato.

```
program restaura
! rifoca un'immagine sfocata con iter=30 iterazioni di Richardson
  implicit none
  integer, parameter    :: dp=kind(0.d0)
  integer               :: m, n, i, j, k, liv, iter
  real(dp), allocatable :: a(:,:), b(:,:), x(:,:), psf(:,:)
  integer               :: aux
  character(len=2)      :: ch
  write(*,*)"assegna il numero di iterazioni di Richardson"
  read(*,*)iter
! legge la psf dal file psf.txt
  open(unit=10, file='psf.txt', action='read')
  read(10,*) k
  allocate(psf(-k:k,-k:k))
  do i=-k,k
    do j=-k,k
      read(10,*) psf(i,j)
    end do
  end do
! legge l'immagine sfocata
  open(unit=20, file='sfocata.pgm',action='read')
  read(20,*)ch
  read(20,*) n, m
  read(20,*) liv
  allocate(b(1:m,1:n))
  do i=1,m
    do j=1,n
      read(20,*) b(i,j)
    enddo
  enddo
! rifoca l'immagine
  allocate(a(1-k:m+k,1-k:n+k))
  call gc(m,n,k,iter,psf,b,x)
! moltiplica calcola a=H^Tx
  call atx(m,n,k,psf,x,a)
! scrive l'immagine rifocata
  open(unit=30, file='rifocata.pgm', action='write')
  write(30,'(A)') ch
  write(30,*) n+2*k,m+2*k
  write(30,*) liv
  do i=1-k,m+k
    do j=1-k,n+k
```

Figura 16: Gradiente coniugato dopo 20 iterazioni in presenza di rumore

```
      aux=a(i,j)
      aux=max(0,aux)
      aux=min(255,aux)
      write(30,*)aux
    enddo
  enddo
end program restaura
```

La granulosità che compare nell'immagine è data dall'amplificazione del rumore di quantizzazione generato dall'aver registrato, troncando al valore intero, i pixel dell'immagine sfocata.

Questo comportamento di alcuni metodi iterativi viene detto *regolarizzante*. Nel caso del metodo di Richardson, questo comportamento si riesce a dimostrare facilmente. Osserviamo che il sistema che viene risolto dal metodo di Richardson ha una matrice simmetrica e (semi)definita positiva. Quindi i suoi valori singolari coincidono con gli autovalori e i suoi vettori singolari con gli autovettori. La matrice di iterazione del metodo di Richardson è data da $P = I - \theta S$ dove $S = HH^T$. Se $0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$ sono gli autovalori di S, dove N è la dimensione di S, allora gli autovalori di P sono $1-\theta\lambda_1 \geq 1-\theta\lambda_2 \geq \cdots \geq 1-\theta\lambda_N$, che sono maggiori o uguali a zero perché abbiamo scelto $\theta \leq 1/\|H\| \leq 1/\lambda_N$. Indicando con $u_1, \ldots, u_N$ i corrispondenti autovettori di S e di P, si ha

$$P = \sum_i (1 - \theta\lambda_i) u_i u_i^T.$$

Figura 17: Gradiente coniugato dopo 40 iterazioni in presenza di rumore

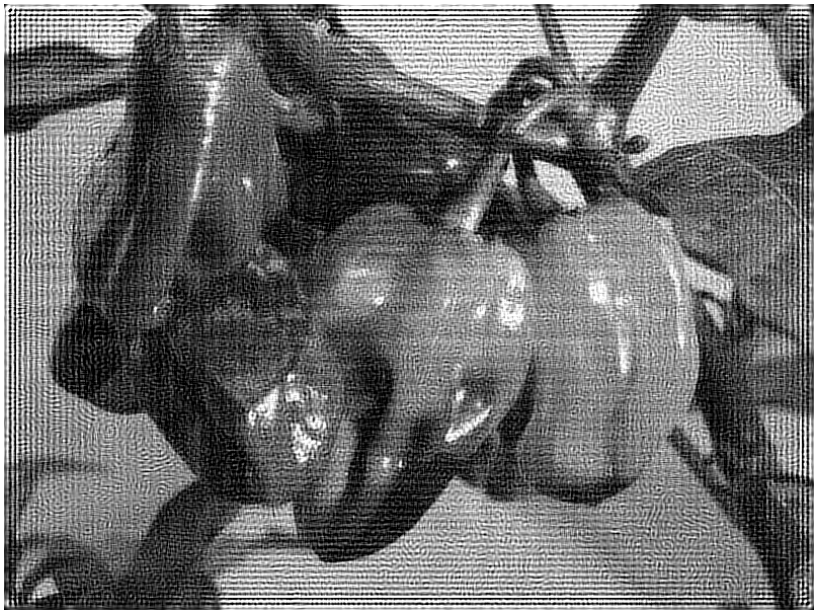

Figura 18: Gradiente coniugato dopo 80 iterazioni in presenza di rumore

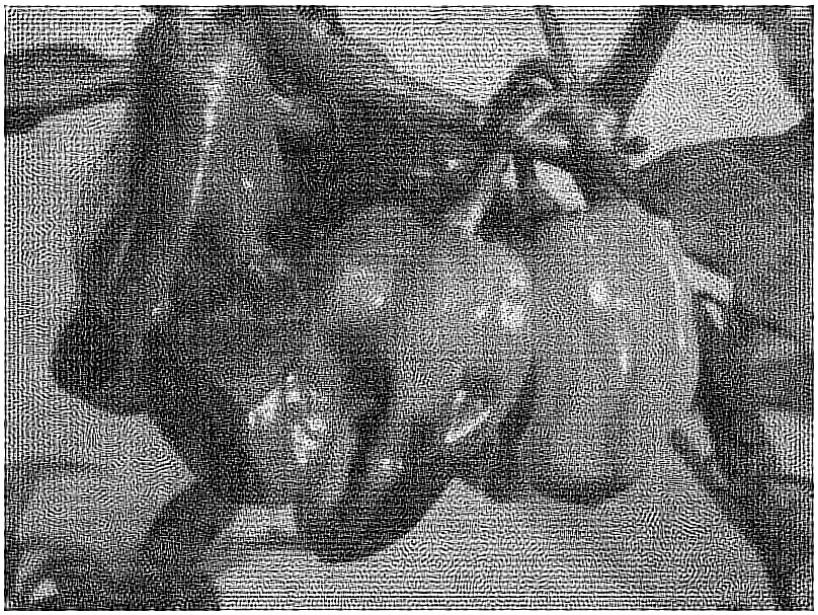

Figura 19: Gradiente coniugato dopo 160 iterazioni in presenza di rumore

Poiché nel metodo di Richardson l'errore e_ν dopo ν passi è dato da $P^\nu e_0$ e poiché

$$P^\nu = \sum_i (1 - \theta\lambda_i)^\nu u_i u_i^T,$$

ne segue che

$$e_\nu = P^\nu e_0 = \sum_i (1 - \theta\lambda_i)^\nu (u_i^T e_0) u_i.$$

Cioè a convergere più rapidamente a zero sono le componenti dell'errore lungo le direzioni u_i per cui $1 - \theta\lambda_i$ prende i valori più piccoli, e quindi λ_i prende i valori più grandi.

In questo modo, la soluzione che si ottiene con un numero contenuto di iterazioni può essere vista come una approssimazione che è precisa solo nelle componenti in frequenza più basse dell'immagine così come accade troncando lo sviluppo in serie di Fourier dell'immagine stessa che genera l'effetto Gibbs. Al crescere del numero di iterazioni vengono approssimate in misura sempre più accurata anche le componenti in alta frequenza e quindi, a lungo andare anche le componenti che contengono il rumore. Il numero di iterazioni diventa allora un parametro importante che determina sia la precisione di rimessa a fuoco dell'immagine, sia il parametro che determina l'insorgere del rumore nella soluzione approssimata.

Data la lentezza di convergenza nello spazio del rumore, il metodo di Richardson richiede molte iterazioni affinché il rumore si manifesti in modo signi-

Figura 20: Richardson dopo 50000 iterazioni in presenza di rumore

ficativo. La figura 20 riporta la approssimazione della soluzione ottenuta dopo 50000 passi del metodo di Richardson.

In mancanza di metodi iterativi regolarizzanti, o in alternativa ad essi, sono state sviluppate tecniche di regolarizzazione ad hoc che permettono di controllare l'amplificazione degli errori presenti nei dati, quale la regolarizzazione di Tikhonov. Questi metodi permettono anche di contenere gli effetti di generazione di echi nelle zone di contrasto netto tipo i bordi delle immagini.

Diamo una descrizione sintetica e parziale di una di queste tecniche.

Il sistema lineare $HH^T x = b$ ha matrice HH^T che è mal condizionata e questo mal condizionamento come si è visto è la causa dell'amplificazione degli errori presenti nel termine noto. Per mitigare il mal condizionamento si sostituisce questo sistema col sistema

$$(HH^T + \alpha I)w = b,$$

dove α è un numero positivo "piccolo". La scelta di α è basata su motivazioni "qualitative" che esprimiamo di seguito in modo informale.

Gli autovalori della matrice $HH^T + \alpha I$ sono chiaramente $\sigma_i^2 + \alpha$ mentre la matrice degli autovettori rimane sempre U. La soluzione calcolata al posto di x diventa

$$w = \sum_i u_i \frac{1}{\sigma_i^2 + \alpha}(u_i^T b)$$

Se, come accade in queste situazioni gli autovalori più piccoli σ_i^2 sono molto vicini a 0 allora una scelta di α significativamente più grande dei σ_i^2 piccoli

Figura 21: Gradiente coniugato regolarizzato con $\alpha = 0.001$ dopo 20 iterazioni in presenza di rumore

produce un valore di $\sigma_i^2 + \alpha$ sufficientemente lontano da 0. In questo modo i termini $u_i \frac{1}{\sigma_i^2+\alpha}(u_i^T b)$ corrispondenti non subiscono una grande amplificazione.

D'alto canto per gli autovalori σ_i^2 più grandi, e quindi molto più grandi di α le quantità σ_i^{-2} e $1/(\sigma_i^2 + \alpha$ non differiscono relativamente di molto essendo la loro distanza relativa $\alpha/(\sigma_i^2(\sigma_i^2 + \alpha))$. Succede quindi che le componenti in "bassa frequenza" della soluzione w non differiscono significativamente dalle componenti in bassa frequenza di x mentre nelle componenti in alta frequenza non viene amplificato il rumore che ivi si annida.

L'efficacia di questa tecnica è mostrata nelle figure 21-24 dove si è scelto un parametro $\alpha = 0.001$.

La convergenza del metodo del gradiente coniugato può essere accelerata usando tecniche di precondizionamento Non diamo dettagli teorici a riguardo, ci limitiamo a riportare la versione del metodo del gradiente coniugato precondizionato.

1. Si fissa $\epsilon > 0$ e si sceglie $x^{(0)}$ a caso
2. $r^{(0)} = bAx^{(0)}$
3. **for** $i = 1, 2, \ldots$
4. risolvi il sistema $Mz^{(i1)} = r^{(i1)}$
5. $\rho_{i1} = r^{(i1)T} z^{(i1)}$

Figura 22: Gradiente coniugato regolarizzato con $\alpha = 0.001$ dopo 40 iterazioni in presenza di rumore

Figura 23: Gradiente coniugato regolarizzato con $\alpha = 0.001$ dopo 80 iterazioni in presenza di rumore

Figura 24: Gradiente coniugato regolarizzato con $\alpha = 0.001$ dopo 160 iterazioni in presenza di rumore

6. **if** $i = 1$
7. $p^{(1)} = z^{(0)}$
8. **else**
9. $\beta_{i1} = \rho_{i1}/\rho_{i2}$
10. $p^{(i)} = z^{(i1)} + \beta_{i1} p^{(i1)}$
11. **end if**
12. $q^{(i)} = Ap^{(i)}$
13. $\alpha_i = \rho_{i1}/p^{(i)T} q^{(i)}$
14. $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$
15. $r^{(i)} = r^{(i1)} \alpha_i q^{(i)}$
16. **if** $\|r^{(i)}\| \leq \epsilon$ arresta le iterazioni
17. **end if**
18. **end for**

In questa formulazione la matrice simmetrica e definita positiva M, detta *precondizionatore* , va scelta in modo che la risoluzione del sistema $Mz = r$ sia computazionalmente facile e che la matrice $M^{-1}A$ abbia il maggior numero possibile di autovalori raggruppato attorno a 1. Cioè M deve rassomigliare spettralmente ad A ma essere al tempo stesso facilmente invertibile.

6 Matrici di Toeplitz

Una matrice viene detta di Toeplitz se i suoi elementi dipendono unicamente dalla differenza degli indici per cui si hanno valori uguali lungo ciascuna diagonale della matrice parallela alla diagonale principale. Matrici di Toeplitz si incontrano in molti problemi provenienti dalle applicazioni dove la struttura Toeplitz riflette l'invarianza per traslazione spaziale o temporale delle grandezze fisiche coinvolte nel modello matematico. Questa classe di matrici ha proprietà spettrali e computazionali di grande interesse. Un'ampia letteratura esiste sulle matrici di Toeplitz che tocca diverse parti teoriche e computazionali della matematica. In questo paragrafo ci soffermiamo su alcuni aspetti computazionali e descriviamo come sia possibile calcolare il prodotto tra una matrice di Toeplitz $n \times n$ ed un vettore in $O(n \log n)$ operazioni aritmetiche.

6.1 Matrici circolanti e FFT

Si consideri il polinomio monico $a(x) = x^n - \sum_{i=0}^{n-1} a_i x^i$ e la matrice companion (di Frobenius) associata

$$C(a) = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ a_0 & a_1 & \dots & a_{n-1} \end{bmatrix}$$

è immediato verificare che se ξ_i è uno zero di $a(x)$, cioè se $a(\xi_i) = 0$, allora $v^{(i)} = (\xi_i^j)_{j=0,\dots,n-1}$ verifica la relazione

$$Fv^{(i)} = \begin{bmatrix} \xi_i \\ \xi_i^2 \\ \vdots \\ \xi_i^n \end{bmatrix} = \xi_i v^{(i)}.$$

Per cui, se $a(x)$ ha n zeri distinti $\xi_1, \dots, \xi_n$ allora la matrice di Vandermonde $V = (\xi_i^{j-1})_{i,j=1,n}$, è tale che

$$V^{-1}FV = \mathrm{Diag}(\xi_1, \dots, \xi_n).$$

Si consideri il polinomio $a(x) = x^n - 1$ e la matrice companion associata

$$C = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ 1 & 0 & \dots & 0 \end{bmatrix}$$

Il polinomio $a(x)$ ha per zeri le radici n-esime dell'unità, $\xi_i = \omega_n^{i-1}$, $i = 1, \dots, n$, dove $\omega_n = \cos(\frac{2\pi}{n}) + \mathtt{i}\sin(\frac{2\pi}{n})$, dove $\mathtt{i}$ è l'unità immaginaria. Per cui, posto $\Omega = (\omega_n^{(i-1)(j-1)})$, e $V = \frac{1}{\sqrt{n}}\Omega$, vale

$$V^H V = I, \quad V^H C V = \mathrm{Diag}(1, \omega_n, \omega_n^2, \dots, \omega_n^{n-1}), \quad V C V^H = \mathrm{Diag}(1, \bar{\omega}_n, \bar{\omega}_n^2, \dots, \bar{\omega}_n^{n-1}),$$

dove V^H indica la matrice trasposta coniugata di V e $\bar{\omega}_n$ il complesso coniugato di ω_n. Si osservi tra l'altro che, essendo ω_n di modulo 1, vale $\bar{\omega}_n = \omega_n^{-1}$.

Si ricorda che Ω è la matrice che definisce la trasformata discreta di Fourier. Infatti, per definizione, l'applicazione $y = \frac{1}{n}\Omega^H x$ è la trasformata discreta di Fourier (DFT), mentre l'applicazione $x = \Omega y$ è la trasformata discreta inversa di Fourier (IDFT). Si ricorda ancora che, se n è una potenza intera di 2, esistono algoritmi per calcolare la DFT e la IDFT in circa $(3/2)n \log_2 n$ operazioni aritmetiche. Se n è un intero positivo qualsiasi esistono comunque metodi per il calcolo della DFT e della IDFT che richiedono $O(n \log n)$ operazioni aritmetiche. Tali algoritmi sono noti come algoritmi FFT, acronimo di Fast Fourier Transform. Esistono diverse implementazioni degli algoritmi FFT. Una delle implementazioni più efficienti è quella della FFTW di Matteo Frigo e Steven G. Johnson. Nel seguito denotiamo rispettivamente con $y = \mathrm{DFT}(x)$ e $x = \mathrm{IDFT}(y)$ la trasformata discreta di Fourier e la sua inversa.

Si osservi che C è una matrice di permutazione associata alla permutazione ciclica $\pi_i = i + 1 \bmod n$ che agisce su $\{0, 1, \dots, n-1\}$. Cioè il generico vettore $x = (x_i)_{i=0,1,\dots,n-1}$ viene trasformato da C in $Cx = (x_1, x_2, \dots, x_{n-1}, x_1)^T$. Per cui anche le matrici C^k, $k = 0, 1, \dots, n-1$ sono matrici di permutazione associate alla permutazione $\pi \circ \pi \circ \cdots \circ \pi$ (k-volte). In particolare risulta che C^k è la matrice che ha elementi uguali a 1 in posizione (i, j) se $j - i = k \bmod n$, ed elementi nulli altrove. Conseguentemente la matrice $C(a) = \sum_{i=0}^{n-1} a_i C^i$ avrà elementi $C(a) = (a_{j-i \bmod n})$. Questa struttura è evidenziata qui sotto nel caso $n = 5$

$$\begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 \\ a_4 & a_0 & a_1 & a_2 & a_3 \\ a_3 & a_4 & a_0 & a_1 & a_2 \\ a_2 & a_3 & a_4 & a_0 & a_1 \\ a_1 & a_2 & a_3 & a_4 & a_0 \end{bmatrix} \tag{4}$$

Matrici siffatte sono chiamate *matrici circolanti*. Ogni matrice circolante $C(a)$ può essere vista come un polinomio nella matrice C con coefficienti dati dagli elementi della prima riga di $C(a)$.

Si osservi che, posto $V = \frac{1}{\sqrt{n}}\Omega$, dalla relazione $VCV^H = \mathrm{Diag}(1, \omega_n, \ldots, \omega_n^{n-1})$ e dal fatto che $C(a) = \sum_{i=0}^{n-1} a_i C^i$ segue che

$$VC(a)V^H = \sum i = 0^{n-1} a_i \mathrm{Diag}(1, \bar{\omega}_n, \bar{\omega}_n^2, \ldots, \bar{\omega}_n^{n-1})^i$$

e quindi

$$C(a) = V^H D V = \frac{1}{n}\Omega^H D \Omega, \quad , D = \mathrm{Diag}(d), \quad d = \bar{\Omega}(a_0, a_1, \ldots, a_{n-1})^T.$$

Poiché la prima riga di $C(a)$ si ottiene invertendo l'ordine degli elementi della prima colonna di $C(a)$, si ha che $d = \Omega C(a) e_1$, dove $e_1 = (1, 0, \ldots, 0)^T$. Per cui possiamo scrivere

$$C(a) = \frac{1}{n}\Omega^H D \Omega, \quad D = \mathrm{Diag}(d), \quad d = \Omega C(a) e_1,$$

così che il prodotto $y = C(a)x$ può essere scritto nella forma

$$y = \frac{1}{n}\Omega^H \left((\Omega C(a) e_1) \odot (\Omega x) \right),$$

dove $\odot$ indica il prodotto componente a componente di due vettori, cioè $u \odot w = (u_i w_i)$. In questo modo il prodotto $y = C(a)x$, dati x ed a, può essere calcolato in modo efficiente dal seguente algoritmo

- $z = \mathrm{IDFT}(x)$
- $w = \mathrm{IDFT}(a_0, a_{n-1}, \ldots, a_1)$
- $u = w \odot z$
- $y = \mathrm{DFT}(u)$

che richiede il calcolo di due trasformate inverse discrete di Fourier e di una trasformata diretta. Se n è potenza intera di 2 il costo computazionale complessivo è di circa $\frac{9}{2} n \log_2 n$ operazioni aritmetiche.

6.2 Prodotto matrice di Toeplitz - vettore

Il prodotto tra una matrice di Toeplitz e un vettore può essere ricondotto al calcolo del prodotto matrice circolante-vettore procedendo nel seguente modo.

Data una matrice di Toeplitz $T = (t_{j-i})$ di dimensione $n \times n$ si consideri la matrice circolante A di dimensione $p\times$ con $p \geq 2n$ la cui prima riga è

$$(t_0, t_1, \ldots, t_{n-1}, 0, \ldots, 0, t_{-n+1}, \ldots t_{-2}, t_{-1})$$

e si osservi che la sottomatrice principale di testa di A di ordine n coincide con T. Si veda ad esempio il caso

$$A = \begin{bmatrix} a_0 & a_1 & a_2 & 0 & a_{-2} & a_{-1} \\ a_{-1} & a_0 & a_1 & a_2 & 0 & a_{-2} \\ a_{-2} & a_{-1} & a_0 & a_1 & a_2 & 0 \\ 0 & a_{-2} & a_{-1} & a_0 & a_1 & a_2 \\ a_2 & 0 & a_{-2} & a_{-1} & a_0 & a_1 \\ a_1 & a_2 & 0 & a_{-2} & a_{-1} & a_0 \end{bmatrix}$$

in cui la sottomatrice principale di testa 3×3 è una generica matrice di Toeplitz.

Per calcolare il prodotto $y = Tx$ si considera il vettore $\hat{x}$ di dimensione p ottenuto estendendo con zeri il vettore x, cioè $\widehat{x}^T = (x^T, 0, \ldots, 0)$. Per il vettore $\widehat{y} = A\widehat{x}$ vale allora

$$\widehat{y} = \begin{bmatrix} y \\ y' \end{bmatrix}$$

dove y' è un opportuno vettore.

In questo modo il calcolo di $y = Ax$ è ricondotto al calcolo di un prodotto matrice circolante-vettore per un costo di $O(n \log n)$. Inoltre, poiché la dimensione p della matrice circolante in cui si immerge la matrice di Toeplitz è arbitraria purché almeno $2n$, possiamo sceglierla pari a una potenza intera di 2 in modo che le trasformate discrete di Fourier possano essere calcolate in modo più efficiente.

6.3 Matrici a blocchi

Nel caso di matrici Toeplitz $n \times n$ a blocchi con blocchi $m \times m$ di Toeplitz è possibile condurre una analisi analoga ed arrivare ad un algoritmo per il calcolo del prodotto matrice-vettore con un costo computazionale di $O(mn \log mn)$ operazioni aritmetiche. Per ottenere questo è utile introdurre la classe delle matrici circolanti a blocchi con blocchi circolanti

$$\mathcal{C} = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} a_{i,j} C_m^i \otimes C_n^j,$$

dove C_m e C_n sono le matrici companion associate rispettivamente ai polinomi $x^m - 1$ e $x^n - 1$, e $\otimes$ indica il prodotto di Kronecker definito nel seguente modo: date due matrici A e B, la matrice $A \otimes B$ è la matrice a blocchi il cui blocco in posizione (i, j) è dato da $a_{i,j}B$. Tutte le matrici di questa classe sono diagonalizzate dalla trasformazione per similitudine definita dalla matrice $V_m \otimes V_n$, dove $V_m = \frac{1}{m}\Omega_m$, $V_n = \frac{1}{n}\Omega_n$ e $\Omega_m = (\omega_m^{(i-1)(j-1)})$, $\Omega_n = (\omega_n^{(i-1)(j-1)})$. Vale infatti

$$(V_m \otimes V_n)(\mathcal{C})(V_m^{-1} \otimes V_n^{-1}) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} a_{i,j} D_m^i \otimes D_n^j,$$

che è una matrice diagonale essendo $D_m = \text{Diag}(1, \bar{\omega}_m, \ldots, \bar{\omega}_m^{m-1})$, $D_n = \text{Diag}(1, \bar{\omega}_n, \ldots, \bar{\omega}_n^{n-1})$.

Questa espressione permette di calcolare il prodotto dii una matrice circolante a blocchi con blocchi circolanti con un numero di operazioni dell'ordine di $O(mn \log mn)$. Lasciamo i dettagli al lettore.

Analogamente al caso delle matrici di Toeplitz trattato in precedenza è possibile verificare che una matrice di Toeplitz a blocchi con blocchi di Toeplitz può essere vista come una sottomatrice principale di una matrice circolante a blocchi con blocchi circolanti. In questo modo il prodotto tra una matrice di Toeplitz a blocchi con blocchi di Toeplitz e un vettore può essere calcolato in $O(mn \log mn)$ operazioni aritmetiche. Lasciamo i dettagli di questo algoritmo al lettore.

In questo contesto si possono costruire dei precondizionatori M, per il gradiente coniugato precondizionato, nella classe delle matrici circolanti a blocchi con blocchi circolanti. Esistono varie modalità per fare questa scelta che si basano sulla teoria spettrale delle matrici di Toeplitz.