Università degli Studi di Cagliari

FACOLTÀ DI INGEGNERIA E ARCHITETTURA

Corso di Laurea Triennale in Ingegneria Elettrica, Elettronica e Informatica

Metodi di regolarizzazione iterativi per il deblurring

Candidato:
Luca Martis

Relatore:
Prof. Giuseppe Rodriguez

Anno Accademico 2020–2021

Alla mia famiglia.

Ringraziamenti

Prima di procedere con la trattazione, vorrei dedicare qualche riga a tutti coloro che mi sono stati vicini in questo percorso di crescita personale e professionale.
Un sentito grazie al prof. Giuseppe Rodriguez per la sua infinita disponibilità e competenza.
Ringrazio i miei genitori e mia sorella per avermi supportato (e soprattutto sopportato) in questo percorso.
Ringrazio i miei nonni e mio zio Sandro che ci sono sempre stati in ogni momento.
Grazie ai miei colleghi per aver rallegrato le lezioni e le pause studio e per avermi assistito nei miei folli dubbi.
Infine grazie ai miei amici e amiche, in particolare i membri della
Buddonsliga, che hanno avuto la pazienza di ascoltare i miei sfoghi e sono sempre riusciti a farmi sorridere anche nei momenti peggiori.

Indice

Capitolo 1

FFT e matrici strutturate

1.1 Trasformata di Fourier

Lo strumento matematico che ci consente di trasferire lo studio dei segnali e dei sistemi dal dominio del tempo al dominio della frequenza è la trasformata di Fourier.

La trasformata di Fourier di una funzione $f(x)$ definita su tutto l'asse reale è la funzione

$$\mathcal{F}(f) = F(k) = \int_{-\infty}^{+\infty} f(x)e^{-jkx}dx,$$

dove j indica l'unità immaginaria, definita nello spazio delle frequenze.
Per trasformata inversa di Fourier di $F(k)$ si intende la funzione

$$\mathcal{F}^{-1}(F) = f(x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(k)e^{jkx}dk$$

definita nello spazio ordinario.

La relazione che definisce la trasformata di Fourier non è direttamente implementabile mediante un elaboratore digitale di segnale, sia perchè richiede l'analisi di segnali continui, $f(x)$ e $F(k)$, sia perchè l'integrale si estende all'infinito e richiederebbe quindi un numero infinito di operazioni.

Per effettuare questa trasformazione mediante calcolatore sono quindi necessarie tre operazioni:

- **Troncamento del segnale**:
- **Campionamento di** $f(x)$
- **Campionamento di** $F(k)$

1.2 DFT

Sia $\mathbf{x} = [x_0, x_1, ..., x_{N-1}]$ il segnale $f(x)$ campionato su un intervallo finito, definiamo la trasformata discreta di Fourier(DFT) di $\mathbf{x}$ come

$$X_k = \sum_{n=0}^{N-1} x_n e^{-jk\frac{2\pi}{N}n}.$$

Sia $\mathbf{X} = [X_1, X_2, ..., X_{N-1}]$, per trasformata discreta inversa di Fourier(IDFT) si intende

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_n e^{jk\frac{2\pi}{N}n}.$$

La DFT può anche essere espressa come moltiplicazione matrice vettore attraverso la matrice di Fourier F_N:

$$\mathbf{X} = F_N \mathbf{x},$$

dove

$$F_N = \begin{bmatrix} \omega_N^{0\cdot 0} & \omega_N^{0\cdot 1} & \cdots & \omega_N^{0\cdot(N-1)} \\ \omega_N^{1\cdot 0} & \omega_N^{1\cdot 1} & \cdots & \omega_N^{1\cdot(N-1)} \\ \omega_N^{2\cdot 0} & \omega_N^{2\cdot 1} & \cdots & \omega_N^{2\cdot(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_N^{(N-1)\cdot 0} & \omega_N^{(N-1)\cdot 1} & \cdots & \omega_N^{(N-1)\cdot(N-1)} \end{bmatrix}, \qquad \omega_N = e^{-\frac{2\pi}{N}j}.$$

Ad esempio, scegliendo $N = 4$

$$F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix},$$

allo stesso modo possiamo esprimere la IDFT come

$$\mathbf{x} = F_N^{-1}\mathbf{X} = \frac{1}{N}F_N^*\mathbf{X}.$$

1.3 FFT

La DFT è uno strumento utile per determinare lo spettro in frequenza di un segnale, ma l'onere computazionale può risultare elevato. Infatti entrambi gli indici i e k devono variare tra 0 e N, pertanto sono necessarie N^2 operazioni di moltiplicazione e addizione. Per questo motivo sono stati sviluppati diversi algoritmi di trasformata rapida di Fourier(Fast Fourier Trasform, FFT) che sfruttando le proprietà di simmetria della DFT ne consentono un calcolo più rapido.
L'algoritmo FFT più diffuso è l'algoritmo di Cooley-Tukey. Grazie a questo possiamo ridurre il costo computazione del calcolo della DFT da $O(N^2)$ a $O(N\ \log N)$.
Per spiegare come funziona la FFT partiamo dal calcolo della DFT di un vettore $\mathbf{x}$ di dimensione N.
Introducendo la notazione $\omega_N^{kn} = e^{-j\frac{2\pi}{N}kn}$ possiamo scrivere:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} x_n \omega_N^{kn}, \qquad k = 0, 1, ..., N-1.$$

L'algoritmo consiste nello spezzare ad ogni passo la sommatoria in due. Separando in termini di indice pari e termini di indice dispari otteniamo:

$$X_k = \sum_{n=0}^{N/2-1} x_{2n}\omega_N^{k(2n)} + \sum_{n=0}^{N/2-1} x_{2n+1}\omega_N^{k(2n+1)} = \sum_{n=0}^{N/2-1} x_{2n}\omega_N^{k(2n)} + \omega_N^k \sum_{n=0}^{N/2-1} x_{2n+1}\omega_N^{k(2n)}.$$

Poichè vale $\omega_N^2 = \omega_{N/2}$ [1] si ha:

$$X_k = \sum_{n=0}^{N/2-1} x_{2n}\omega_{N/2}^{kn} + \omega_N^k \sum_{n=0}^{N/2-1} x_{2n+1}\omega_{N/2}^{kn}.$$

[1] $e^{-j\frac{2\pi}{N/2}} = (e^{-j\frac{2\pi}{N}})^2 = e^{-j\frac{4\pi}{N}}$

Possiamo vedere inoltre che [2] $\omega_{N/2}^{(k+N/2)n} = \omega_{N/2}^{kn}$ e che $\omega_N^{k+N/2} = -\omega_N^k$.

Per $k \geq N/2$ le due sommatorie riprendono gli stessi valori assunti per $0 \leq k \leq N/2$ e l'unico cambiamento si ha nel segno davanti alla seconda sommatoria:

$$\begin{cases} X_k = X_{1k} + \omega_N^k X_{2k}, \qquad k = 0, 1, ..., N/2 - 1. \\ \\ X_{k+N/2} = X_{1k} - \omega_N^k X_{2k}, \qquad k = 0, 1, ..., N/2 - 1. \end{cases}$$

dove X_{1k} e X_{2k} sono rispettivamente le sommatorie associate ai termini pari e dispari del vettore iniziale $\mathbf{x}$.

Quindi la DFT su N campioni può essere vista come combinazione lineare di due DFT su $N/2$ campioni , dove il calcolo di X_{1k} e X_{2k} richiede $(N/2)^2$ operazioni, e il prodotto $\omega_N^k X_{2k}$ richiede $N/2$ operazioni, per un costo complessivo di $2(N/2)^2 + N/2$, che per N abbastanza grandi corrisponde a circa $(N/2)^2$.

Il passo fondamentale della FFT è che possiamo spezzare ad ogni passo le sommatorie, riducendo ad ogni passo il costo computazionale,se $N = 2^n$ possiamo arrivare a spezzare $n - 1$ le sommatorie, fino ad arrivare a un costo minimo di $O(N \log N)$, che per N molto grandi è circa uguale a N.

1.4 Matrici circolanti

Definition 1.4.1. *Una matrice circolante di ordine n è una matrice C i cui elementi soddisfano le relazioni*

$$C_{ij} = c_{i-j}, \qquad i, j = 1, ..., n.$$

$$c_{k-n} = c_k, \qquad k = 1, ..., n - 1.$$

[2] $e^{-j\frac{2\pi}{N/2}(k+N/2)n} = e^{-j\frac{2\pi}{N/2}kn} e^{-j2\,\pi n} = e^{-j\frac{2\pi}{N/2}kn}$

Per esempio una matrice circolante con n=4

$$C = \begin{bmatrix} c_0 & c_3 & c_2 & c_1 \\ c_1 & c_0 & c_3 & c_2 \\ c_2 & c_1 & c_0 & c_3 \\ c_3 & c_2 & c_1 & c_0 \end{bmatrix}.$$

Grazie alle sue proprietà è sufficiente memorizzare solo la prima colonna della matrice C per avere tutte le informazioni che ci servono.

La proprietà principale di una matrice circolante è che può essere diagonalizzata attraverso la matrice di Fourier normalizzata $\mathcal{F} = \frac{1}{\sqrt{N}} F_N$.
Ciò ci permette di fattorizzare una qualunque matrice circolante nella forma

$$C = \mathcal{F}^{-1} \Delta \mathcal{F},$$

dove

$$\Delta = \mathrm{diag}(F_N \mathbf{c})$$

con $\mathbf{c} = (c_0, c_1, c_2, ...c_{n-1})^T$

Inoltre per ogni $\mathbf{x} \in \mathcal{C}^n$

$$\mathcal{F}\mathbf{x} = \frac{1}{\sqrt{n}} \cdot \mathrm{fft}(x), \qquad \mathcal{F}^{-1}\mathbf{x} = \sqrt{n} \cdot \mathrm{ifft}(x).$$

Utilizzando queste proprietà possiamo scrivere il prodotto tra una matrice circolante C e un vettore $\mathbf{x}$ come:

$$C\mathbf{x} = \mathcal{F}^{-1} \Delta \mathcal{F}\ \mathbf{x} = \mathrm{ifft}\ (\mathrm{fft}(\mathbf{c}) \circ \mathrm{fft}(\mathbf{x})) = \mathrm{ifft}(\boldsymbol{\delta} \circ fft(\mathbf{x})),$$

dove $\mathbf{u} \circ \mathbf{v} = (u_1 v_1, ..., u_n v_n)^T$ denota il prodotto elemento per elemento tra due vettori.

Questo prodotto richiede in generale il calcolo di 3 fft e un prodotto elemento per elemento con un costo computazione di $O(nlog(n))$, dove n è la dimensione della matrice, che risulta essere inferiore al costo computazionale di un generico prodotto matrice vettore che richiede un costo computazionale di $O(n^2)$.

1.4.1 Matrici circolanti bi-indice

La definizione di matrice circolante nel caso bi-indice non varia rispetto al mono-indice a patto di considerare gli indici i, j degli elementi della matrice come vettori di due componenti $[n_1, n_2]$.

Possiamo comunque rappresentarla come una matrice mono-indice, attraverso una matrice circolante a blocchi.

Per la memorizzazione non sarà più sufficiente memorizzare un vettore di dimensione n ma dovremo memorizzare una matrice di dimensione $n \times n$, chiamata gnomone, contenente in ciascuna colonna la prima colonna di ciascun blocco della matrice. Per esempio la matrice circolante a blocchi C che corrisponde allo gnomone $C.c$ sarà:

. $$C.c = \begin{bmatrix} 3 & 5 \\ 4 & 6 \end{bmatrix}, \qquad C = \begin{bmatrix} 3 & 4 & 5 & 6 \\ 4 & 3 & 6 & 5 \\ 5 & 6 & 3 & 4 \\ 6 & 5 & 4 & 3 \end{bmatrix}.$$

Analogamente per le matrici circolanti mono-indice, possiamo sfruttare la trasformata di Fourier per calcolare i prodotti matrice vettore, con la differenza che al posto di calcolare la FFT della prima colonna della matrice mono-indice, calcoleremo la FFT in due dimensioni dello gnomone della matrice a blocchi.

1.5 Matrici di Toeplitz

Definition 1.5.1. *Una matrice di Toeplitz di ordine n è una matrice T i cui elementi sono costanti sulle diagonali*

$$T_{ij} = t_{i-j}, \qquad i, j = 1, ..., n.$$

Per esempio, se n=4

$$T = \begin{bmatrix} t_0 & t_{-1} & t_{-2} & t_{-3} \\ t_1 & t_0 & t_{-1} & t_{-2} \\ t_2 & t_1 & t_0 & t_{-1} \\ t_3 & t_2 & t_1 & t_0 \end{bmatrix}.$$

A differenza di una matrice circolante occorre memorizzare sia la prima colonna che la prima riga della matrice e anche le dimensioni della matrice, per avere tutte le informazioni su di essa.

Anche un prodotto Toeplitz-vettore può essere calcolato velocemente. L'idea chiave è quella di inserire la matrice di Toeplitz dentro una matrice circolante C_T :

$$C_T = \left[\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array}\right],$$

dove gli elementi C_{ij} sono matrici e in particolare $C_{11} = C_{22} = T$.

Per esempio utilizzando la matrice precedente con n=4 possiamo scrivere

$$C_T = \left[\begin{array}{cccc|cccc} t_0 & t_{-1} & t_{-2} & t_{-3} & 0 & t_3 & t_2 & t_1 \\ t_1 & t_0 & t_{-1} & t_{-2} & t_{-3} & 0 & t_3 & t_2 \\ t_2 & t_1 & t_0 & t_{-1} & t_{-2} & t_{-3} & 0 & t_3 \\ t_3 & t_2 & t_1 & t_0 & t_{-1} & t_{-2} & t_{-3} & 0 \\ \hline 0 & t_3 & t_2 & t_1 & t_0 & t_{-1} & t_{-2} & t_{-3} \\ t_{-3} & 0 & t_3 & t_2 & t_1 & t_0 & t_{-1} & t_{-2} \\ t_{-2} & t_{-3} & 0 & t_3 & t_2 & t_1 & t_0 & t_{-1} \\ t_{-1} & t_{-2} & t_3 & 0 & t_3 & t_2 & t_1 & t_0 \end{array}\right].$$

Scrivendo la matrice T in questo modo possiamo eseguire il prodotto $T\mathbf{x}$ in maniera più veloce. Prima costruiamo il vettore $\bar{\mathbf{x}}$ aggiungendo zero al vettore $\mathbf{x}$ per arrivare alla dimensione di C_T. Dopo calcoliamo il prodotto $C_T\bar{\mathbf{x}}$, dove poichè C_T è circolante possiamo applicare l'algoritmo con la fft e in seguito estraiamo il risultato che ci serve.

$$C_T\bar{x} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} x \\ 0 \end{bmatrix} = \begin{bmatrix} C_{11}x \\ C_{12}x \end{bmatrix} = \begin{bmatrix} Tx \\ * \end{bmatrix}.$$

1.5.1 Matrice di Toeplitz bi-indice

Passando al bi-indice, a patto di considerare gli indici i, j della matrice come vettori con due componenti le definizioni non cambiano. Allo stesso modo delle matrici circolanti bi-indice, possiamo ricondurre la matrice di Toeplitz

bi-indice a una matrice mono-indice a blocchi chiamata Block-Toeplitz with Toeplitz Block (BTTB).

Questa sarà come una matrice di Toeplitz in cui ogni elemento T_k è una matrice di Toeplitz

$$K = \begin{bmatrix} T_0 & T_{-1} & \cdot & \cdot & \cdot & \cdot & T_{-n+1} \\ T_1 & T_0 & T_{-1} & \cdot & \cdot & \cdot & \cdot \\ T_2 & T_1 \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & T_{-1} \\ T_{n-1} & \cdot & \cdot & \cdot & \cdot & T_1 & T_0 \end{bmatrix}.$$

Per memorizzarla sarà necessaria una matrice contenente la prima riga e la prima colonna di ciascun blocco e anche le dimensioni della matrice.

Anche per questo tipo di matrici strutturate possiamo sfruttare la FFT per velocizzare i calcoli come abbiamo fatto per le matrici di Toeplitz, con la differenza che dovremmo utilizzare la FFT in due dimensioni

1.5.2 Convoluzione

Siano K e f due funzioni integrabili in $\mathbb{R}$. Ad esse è associabile la funzione, anch'essa integrabile in $\mathbb{R}$

$$(K * f) = \int_{-\infty}^{+\infty} K(x-y)f(y)dy = g(x)$$

definita convoluzione della K e della f, nel senso della trasformata di Fourier.

In forma discreta diventa

$$\sum_{y}^{N} K[x-y] \cdot f[y] = g[x].$$

É immediato notare che il primo fattore della sommatoria dipende solo dalla differenza $x-y$, utilizzando gli indici $(i-j)$ al posto di (x, y) la sommatoria

diventa

$$\sum_{j-1}^{N} K_{i-j} \cdot f_j = g_i,$$

quindi:

$$\begin{cases} g_1 = k_0 \cdot f_1 + k_{-1} \cdot f_2 + k_{-2} \cdot f_3 + \cdots k_{1-N} \cdot f_N \\ g_2 = k_1 \cdot f_1 + k_0 \cdot f_2 + k_{-1} \cdot f_3 + \cdots \ k_{2-N} \cdot f_N \\ g_3 = k_2 \cdot f_1 + k_1 \cdot f_2 + k_0 \cdot f_3 + \cdots \ k_{3-N} \cdot f_N \\ \vdots \\ g_N = k_{N-1} \cdot f_1 + k_{N-2} \cdot f_2 + k_{N-3} \cdot f_3 + \cdots k_0 \cdot f_N \end{cases}$$

Usando la notazione matriciale :

$$\begin{bmatrix} k_0 & k_{-1} & k_{-2} & \cdots & k_{1-N} \\ k_1 & k_0 & k_{-1} & \cdots & k_{2-N} \\ k_2 & k_1 & k_0 & \cdots & k_{2-N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ k_{N-1} & k_{N-2} & k_{N-3} & \cdots & k_0 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_N \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ \vdots \\ g_N \end{bmatrix}.$$

Quindi possiamo scrivere l'operazione di convoluzione discreta come prodotto di una matrice per un vettore :

$$K\mathbf{f} = \mathbf{g},$$

dove si vede facilmente che K è una matrice di Toeplitz.

Utilizzando le proprietà di tale matrice strutturata possiamo diminuire il costo computazionale di tale operazione passando da un costo pari a $O(N^2)$ a un costo di $O(N \ \log N)$.

1.5.3 Convoluzione in due dimensioni

Se il segnale di ingresso è bidimensionale il modello matematico è

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} K(x-z,y-t)f(z,t)dzdt = g(x,y),$$

che in forma discreta diventa

$$\sum_r \sum_s K_{i-r,j-s}\ f_{r,s} = g_{i,j}.$$

Sviluppando la sommatoria otteniamo :

$$g_{ij} = \sum_r k_{i-r,j}f_{r0} + k_{i-r,j-1}f_{r1} + \cdots + k_{i-r,j-i}f_{ri} + \cdots k_{N-1-r,j-N-1}f_{r,N-1},$$

osservando meglio i singoli elementi della sommatoria vediamo che questi possiamo ricondurli a un operazione di moltiplicazione di matrice di Toeplitz per un vettore colonna

$$\sum_r k_{i-r,j}f_{r0} = T_j f_{r0},$$

quindi il generico vettore colonna G_j possiamo scriverlo come

$$G_j = T_{j-0}F_0 + T_{j-1}F_1 + T_{j-2}F_2 + \cdots T_{j-n}F_N = \sum_s T_{j-s}F_s,$$

che è molto simile al risultato che avevamo ottenuto in una dimensione, con la differenza che in questo caso F indica un vettore colonna della matrice f e T indica una matrice di Toeplitz.

$$\begin{cases} G_1 = T_0 \cdot F_1 + T_{-1} \cdot F_2 + k_{-2} \cdot F_3 + \cdots T_{1-N} \cdot F_N \\ G_2 = T_1 \cdot F_1 + T_0 \cdot F_2 + k_{-1} \cdot F_3 + \cdots\ T_{2-N} \cdot F_N \\ G_3 = T_2 \cdot F_1 + T_1 \cdot F_2 + K_0 \cdot F_3 + \cdots\ T_{3-N} \cdot F_N \\ \vdots \\ G_N = T_{N-1} \cdot F_1 + T_{N-2} \cdot F_2 + T_{N-3} \cdot F_3 + \cdots T_0 \cdot F_N \end{cases}$$

Usando la notazione matriciale :

$$\begin{bmatrix} T_0 & T_{-1} & \cdot & \cdot & \cdot & \cdot & T_{1-N} \\ T_1 & T_0 & T_{-1} & \cdot & \cdot & \cdot & T_{2-N} \\ T_2 & T_1 \cdot & \cdot & \cdot & \cdot & \cdot & T_{3-N} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ T_{N-1} & \cdot & \cdot & \cdot & \cdot & T_1 & T_0 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ \vdots \\ F_N \end{bmatrix} = \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ \vdots \\ G_N \end{bmatrix}.$$

Possiamo quindi ricondurre K a una matrice BTTB e trasformando la funzione f in un vettore colonna composto dalle colonne della matrice f possiamo scrivere l'operazione di convoluzione come un prodotto matriciale :

$$K\mathbf{f} = \mathbf{g},$$

dove anche in questo caso possiamo utilizzare le proprietà della matrice strutturata K per ridurre il costo computazionale.

Capitolo 2

Metodi di regolarizzazione

2.1 Problemi mal posti

Un problema è **ben posto** se esso possiede, in un prefissato campo di definizione, una e una sola soluzione e questa dipende con continuità dai dati. In caso contrario viene detto **mal posto.**

Quando il terzo vincolo viene violato può accadere che una piccola variazione sui dati possa portare a una soluzione molto differente da quella corrispondente ai dati esatti.

La misura quantitativa di come una soluzione viene influenzata dalle peturbazioni sui dati, quindi di quanto l'errore sui dati possa essere amplificato nel risultato, viene detta **condizionamento del problema**.

Si definisce **numero di condizione** di una matrice A, relativamente alla risoluzione di un sistema lineare, la quantità:

$$k(A) = ||A||\ ||A^{-1}||.$$

Il numero di condizione misura il massimo fattore di amplificazione dell'errore relativo sulla soluzione indotta dall'errore relativo sui dati.

La difficoltà nel risolvere problemi con un numero di condizionamento elevato, è che a causa della presenza di errori nei dati misurati l'utilizzo di metodi

non adatti (come Gauss) porta a dei risultati completamente errati.

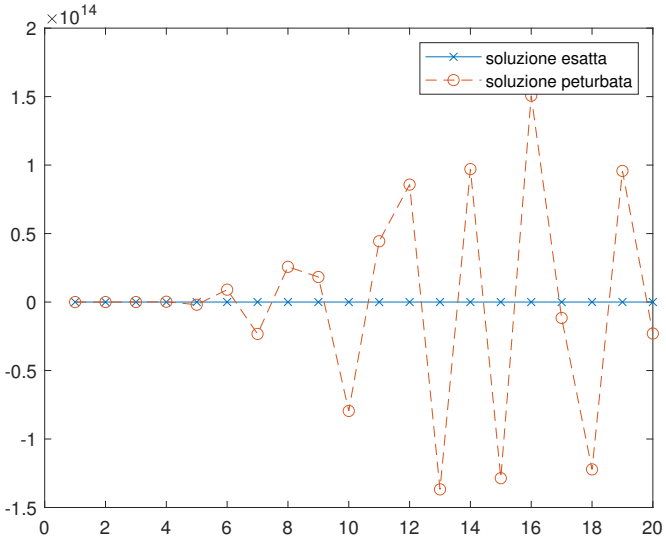


Figura 2.1: Soluzione calcolata con Gauss per una matrice di Hilbert di dimensione 20x20

Si vede dalla figura che utilizzando Gauss per calcolare un sistema di equazioni lineare con numero di condizione elevato (circa 10^{18}) e dati affetti da errore, calcolato come $(10^{-2}) * \text{randn}$, si ottengono delle soluzioni che si discostano dalla soluzione reale di 14 ordini di grandezza.

Il trattamento numerico di sistemi di equazioni lineari con una matrice dei coefficienti A fortemente mal condizionata dipende dal tipo di tipo di condizionamento di A. Esistono due importanti classi di problemi da considerare:

- **Rank-deficent problems**: sorgono dalla presenza di informazioni ridondanti nella matrice A. Sono caratterizzati dall'avere un gruppo di piccoli valori singolari separati da un ben preciso gap dai valori singolari più grandi. Per risolvere numericamente questi problemi è necessario estrarre da A le informazioni linearmente indipendenti e arrivare a un altro problema ben condizionato.

- **Discrete ill-posed problems**: sorgono dalla discretizzazioni di problemi come le equazioni integrali di Fredholm del primo tipo. I valori singolari di A decadono a zero gradualmente senza particolari gap tra loro. Per questo tipo di problemi l'obbiettivo è quello di trovare un bilanciamento tra la norma residua e la dimensione della soluzione che ci si aspetterebbe.

Prima si pensava che i problemi mal posti fossero artificiali, nel senso che non potessero rappresentare un sistema fisico. Adesso invece sappiamo che numerosi problemi mal posti si presentano in diverse aree della scienza e dell' ingegneria. In particolare sono importanti quei problemi che vengono definiti **problemi inversi**. Possiamo rappresentare un problema lineare con la formula generale :

$$\int_{\Omega} \text{input} \times \text{system}\ d\Omega = \text{output}.$$

In questa formulazione, il **problema diretto** consiste nel calcolare l'output, avendo a disposizione l'input e la descrizione matematica del sistema, invece l'obbiettivo di un **problema inverso** è di determinare l'input a partire dal sistema e dalla misurazione dell'output.

2.2 Fattorizzazione SVD

Uno strumento di analisi importante per problemi malcondizionati è la fattorizzazione SVD.

Sia $A \in R^{m\times n}$ con $m > n$. La SVD (singular value decomposition) di A è la decomposizione nella forma

$$A = U\Sigma V^T = \sum_{i=n}^{n} \mathbf{u_i}\sigma_i\mathbf{v_i^T},$$

dove $U = (\mathbf{u_1}, ..., \mathbf{u_n}) \in R^{m\times n}$ e $V = (\mathbf{v_1}, ..., \mathbf{v_n}) \in R^{n\times n}$ sono matrici con colonne ortonormali e dove $\Sigma = diag(\sigma_1, ..., \sigma_n)$ ha elementi non negativi disposti in ordine decrescente

$$\sigma_1 \geq ... \geq \sigma_n \geq 0.$$

I numeri σ_i sono detti valori singolari di A e sono uguali alla radice quadrata degli autovalori delle matrice AA^T e A^TA. Dalle relazioni $A^TA = V\Sigma^2V^T$ e $AA^T = U\Sigma^2U^T$ troviamo che la SVD di A è fortemente legata alla diagonalizzazione delle matrici simmetriche semidefinite positive A^TA e AA^T. Inoltre possiamo anche vedere che il numero di condizione di A è dato dal rapporto σ_1/σ_n.

Inoltre per quanto riguarda i problemi discreti mal posti si trova spesso che :

- I valori singolari σ_i decadono a zero senza particolari separazioni e all'aumentare della dimensione di A aumenta il numero di valori singolari piccoli
- I vettori singolari sinistri $\mathbf{u_i}$ e destri $\mathbf{v_i}$ tendono ad avere più cambi di segni nei loro elementi all'aumentare dell'indice i

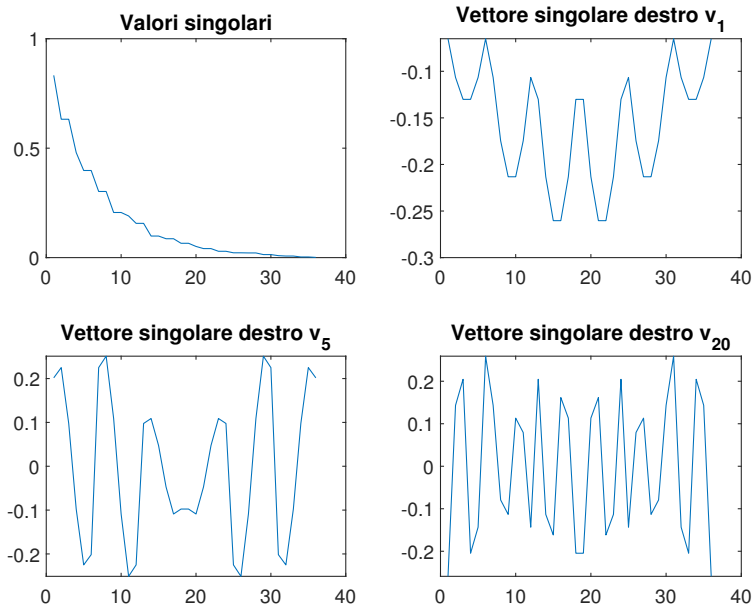


Figura 2.2: SVD di una BTTB 36x36

Considerando quindi Ax di un vettore arbitrario x, usando la SVD otteniamo

$$A\mathbf{x} = \sum_{i=n}^{n} \sigma_i(\mathbf{v_i^T x})\mathbf{u_i},$$

che mostra chiaramente che grazie alla moltiplicazione per σ_i le componenti ad alta frequenza di x sono più smorzate delle componenti a bassa frequenza.

Al contrario il problema inverso ha l'effetto opposto: amplifica le componenti ad alta frequenza del vettore b, e poichè il rumore è un segnale ad alta frequenza, amplifica enormemente il rumore.

2.3 Regolarizzazione

La primaria difficoltà legata ai problemi mal posti è che sono in pratica indeterminati a causa dei piccoli valori singolari di A. É quindi necessario incorporare ulteriori informazioni sulla soluzione desiderata per stabilizzare il problema e isolare una soluzione utile. Questo è l'obbiettivo dei metodi detti di **regolarizzazione**.

Da una prospettiva generale possiamo essenzialmente scegliere due classi di metodi di regolarizzazione:

- **Metodi diretti**: sono basati sulla decomposizione della matrice in forme canoniche ,come per esempio la SVD, e il costo computazionale può essere calcolato a priori.

- **Metodi iterativi**: sono basati su schemi iterativi che accedono alla matrice A e A^T solo per eseguire operazioni di matrice per vettore e producono una sequenza di vettori $x^{(k)}, k = 1, 2...$, che converge alla soluzione desiderata. In questi metodi il parametro di regolarizzazione è il numero di iterazioni k e ciò che si cerca è una **semiconvergenza**, questo poichè il vettore $x^{(k)}$ all'inizio converge verso una soluzione regolarizzata, ma all'aumentare della soluzione converge verso una soluzione indesiderata.

2.4 Metodi diretti

2.4.1 TSVD

Considerando il sistema $Ax = b$, applichiamo la SVD alla matrice A

$$U\Sigma V^T x = b,$$

essendo le matrici U e V ortogonali ed essendo Σ diagonale è semplice invertirle e quindi possiamo calcolare x come

$$x = V\Sigma^{-1}U^T\ b,$$

che possiamo anche scrivere come

$$x = \sum_{i=1}^{n} \frac{\mathbf{u_i^T b}}{\sigma_i}\mathbf{v_i}.$$

Si vede chiaramente che la soluzione del sistema è dominata dai valori nella somma che corrispondono ai più piccoli valori σ_i.

La TSVD(truncated SVD) consiste nel troncare la sommatoria a un certo valore k scartando le componenti dominate dal rumore.

$$\mathbf{x_k} = \sum_{i=1}^{k} \frac{\mathbf{u_i^T b}}{\sigma_i}\mathbf{v_i}, \qquad k < n.$$

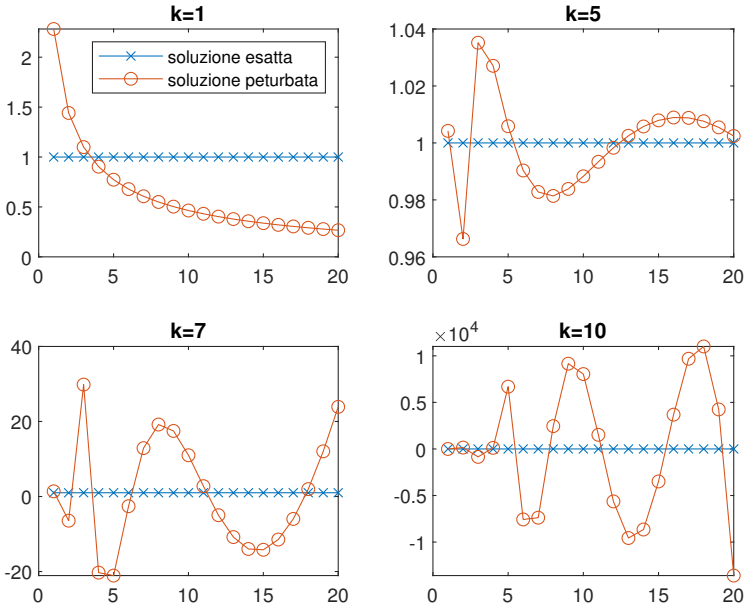


Figura 2.3: Risultati calcolati con la TSVD per una matrice di Hilbert di dimensione 20x20

Si vede dalla figura come aumentando il valore di k ci si avvicini sempre di più alla soluzione esatta, fino ad arrivare al massimo della precisione con $k = 5$, aumentando k ulteriormente torna a dominare la componente del rumore e ci si discosta sempre di più dalla soluzione esatta sino ad arrivare alla soluzione trovata con Gauss.

2.5 Metodi iterativi

Un grande vantaggio di questo tipo di metodi è che si applicano molto bene a problemi di grandi dimensioni. Questo perchè :

- La matrice A conserva le sue proprietà, quindi si possono velocizzare i calcoli se questa è strutturata o sparsa.
- Le operazioni possono essere facilmente parallelizzate.
- A ogni iterazione viene calcolato sia $x^{(k)}$ sia il suo residuo $r^{(k)} = b - Ax^{(k)}$ che è di aiuto per decidere quando fermare le iterazioni.

2.5.1 Metodo del gradiente coniugato

L'algoritmo del gradiente coniugato è un metodo iterativo per risolvere sistemi lineari che hanno matrice dei coefficienti simmetrica semi definita positiva. In particolare il metodo è applicato alle equazioni normali $A^TA\mathbf{x} = A^T\mathbf{b}$ in cui la matrice A^TA è simmetrica semi definita positiva.

Ciò che fa l'algoritmo è trovare il punto di minimo della forma quadratica

$$Q(\mathbf{x}) = \frac{1}{2}\ \mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b}.$$

Infatti, grazie alle proprietà di A:

$$\nabla Q(\mathbf{x}) = A\mathbf{x} - \mathbf{b},$$

da cui

$$\nabla Q(\mathbf{x}) = 0 \Leftrightarrow A\mathbf{x} = \mathbf{b}.$$

Quindi se noi troviamo il vettore $\mathbf{x}$ che minimizza la funzione $Q(\mathbf{x})$ stiamo trovando il punto in cui si annulla il gradiente di tale funzione e di conseguenza troviamo anche la soluzione del sistema $A\mathbf{x} = \mathbf{b}$.

Un modo per implementare l'algoritmo prevede che partendo da una soluzione iniziale $\mathbf{x_0}$ ci si muova con un passo α_k verso una direzione $\mathbf{p_k}$ che minimizza $Q(\mathbf{x})$:

$$\mathbf{x^{(k)}} = \mathbf{x^{(k-1)}} + \alpha_k \mathbf{d^{(k-1)}},$$

dove

$$\alpha_k = ||A^T\mathbf{r^{(k-1)}}||_2^2 \ / \ ||A\mathbf{d^{(k-1)}}||_2^2,$$
$$\mathbf{r^{(k)}} = \mathbf{r^{(k-1)}} - \alpha_k A\mathbf{d^{(k-1)}},$$
$$\beta_k = ||A^T\mathbf{r^{(k)}}||_2^2 \ / \ ||A^T\mathbf{r^{(k-1)}}||_2^2,$$
$$\mathbf{d^{(k)}} = A^T\mathbf{r^{(k)}} - \beta_k \mathbf{d^{(k-1)}}.$$

Un'importante proprietà del metodo è che la norma del residuo $||\mathbf{r}^{(k)}|| =$

$||\mathbf{b} - A\mathbf{x}^{(k)}||_2$ decresce monotonamente con k e al contrario la norma della soluzione $||\mathbf{x}^{(k)}||_2$ cresce monotonamente con k. Il comportamento monotono di $||\mathbf{r}^{(k)}||$ e $||\mathbf{x}^{(k)}||$ è molto importante in connessione con il criterio della L-curve per la scelta del parametro di stop per il metodo.

Inoltre il metodo produce vettori di iterazione le cui componenti spettrali associate ad autovalori più grandi convergono più rapidamente rispetto alle componenti rimanenti. Lo stesso comportamento si nota quando l'algoritmo è applicato alle equazioni normali e poichè gli autovalori di A^TA sono i quadrati dei valori singolari σ_i, questo significa che le componenti della SVD associate ai valori singolari più grandi tendono a convergere più rapidamente delle altre componenti, in questo caso perciò il metodo ha un intrinseco effetto di regolarizzazione se stoppato prima di arrivare alla convergenza della soluzione del problema ai minimi quadrati.

2.5.2 Bidiagonalizzazione di Lanczos

Il metodo a partire da un vettore iniziale b produce ad ogni passo una matrice bidiagonale $B_k \in R^{(k+1)\times k}$ e due matrici con colonne ortonormali $U_{k+1} \in R^{m\times(k+1)}$ e $V_k \in R^{n\times k}$, tali che

$$AV_k = U_{k+1}B_k, \qquad \mathbf{u_1} = \beta_1^{-1}\mathbf{b},$$

segue che $U_{k+1}^T b = \beta_1 \mathbf{e_1}^{(k+1)}$, dove $\mathbf{e_1}^{(k+1)} = (1, 0, ..., 0)^T$ ha lunghezza $k+1$. I vettori vengono inizializzati con

$$\beta_1 = ||\mathbf{b}||_2, \quad \mathbf{u_1} = \mathbf{b}/\beta_1, \quad \mathbf{v_0} = 0$$

e ad ogni passo vengono calcolati

$$\mathbf{p}^{(k)} = A^T\mathbf{u_k} - \beta_k\mathbf{v_k},$$
$$\alpha_k = ||\mathbf{p}^{(k)}||_2,$$
$$\mathbf{v_k} = \mathbf{p}^{(k)}/\alpha_k,$$
$$\mathbf{q}^{(k)} = A\mathbf{v_k} - \alpha_k\mathbf{u_k},$$
$$\beta_{k+1} = ||\mathbf{q}^{(k)}||_2,$$
$$\mathbf{u_{k+1}} = \mathbf{q}^{(k)}/\beta_{k+1}.$$

La matrice B_k è data da

$$B_k = \begin{bmatrix} \alpha_1 & 0 & & \cdots & 0 \\ \beta_2 & \alpha_2 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & & \ddots & \alpha_k \\ 0 & \cdots & & 0 & \beta_{k+1} \end{bmatrix}.$$

Il vettore $\mathbf{x}^{(k)}$ è calcolato come $V_k\xi^{(k)}$, dove $\xi^{(k)} \in R^k$ risolve il problema $\min ||B_k\xi^{(k)} - U_{k+1}^T b||_2$.

Gli algoritmi lsqr_b e cgls sono matematicamente equivalenti, infatti in aritmetica esatta generano al k-esimo passo lo stesso vettore $\mathbf{x}^{(k)}$. Per problemi mal condizionati a rango pieno che non richiedono regolarizzazione è stato sperimentalmente trovato che lsqr_b sia superiore a cgls e altri metodi. Per problemi discreti mal posti, dove le iterazioni sono stoppate molto prima della convergenza finale, lsqr_b non da soluzioni significativamente migliori di cgls.

2.5.3 Tecnica di ricostruzione algebrica

La tecnica di ricostruzione algebrica conosciuta anche come metodo di Kaczmarz, è un metodo iterativo che accede alla matrice dei coefficenti A una riga alla volta.
Il metodo è equivalente all'algoritmo di Gauss-Seidel applicato al problema $\mathbf{x} = A^T\mathbf{y}, AA^T\mathbf{y} = \mathbf{b}$.

$$\mathbf{x^{k+1}} = \mathbf{x^k} + \omega \, \frac{b_i - \mathbf{a_i^T x}}{||\mathbf{a_i}||^2}\mathbf{a_i}, \qquad i = 1, ..., m,$$

dove A è una matrice $m \times n$ tipicamente sparsa, b_i è la i-esima componente del vettore $\mathbf{b}$, $\mathbf{a_i}$ è la i-esima riga della matrice A e ω è un parametro di rilassamento che può essere costante o decrescere con le interazioni.

Capitolo 3

Problemi unidimensionali

Per la sperimentazione numerica in una dimensione sono stati utilizzati i problemi di test forniti dal 'Regularization tool'. Grazie alle dimensioni sufficientemente piccole delle matrici è stato possibile lavorare con le matrici complete e non è stato necessario ottimizzare i metodi di regolarizzazione.

3.1 Problem test: shaw

Questo problema utilizza un equazione integrale di Fredholm del primo ordine

$$\int_{-\pi/2}^{+\pi/2} K(s,t)f(t)dt = g(s), \qquad -\pi/2 < s < \pi/2,$$

dove K è la PSF di un di una fessura infinitamente lunga e con larghezza pari alla lunghezza d'onda

$$K(s,t) = (\cos(s) + \cos(t))^2 \left(\frac{\sin(u)}{u}\right)^2,$$

$$u = \pi(\sin(s) + \sin(t)),$$

f rappresenta l'intensità della sorgente di luce in funzione dell'angolo di incidenza t

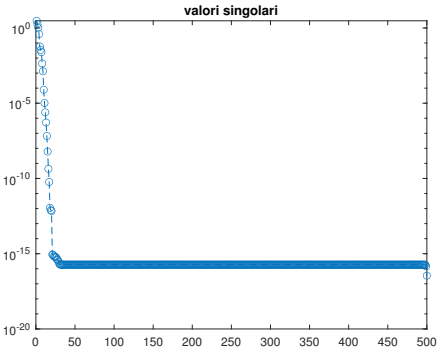
$$f(t) = a_1 \exp(-c_1(t - t_1)^2) + a_2 \exp(-c_2(t - t_2)^2),$$

mentre g è l'intensità della luce osservata nell'immagine prodotta dall'altro lato della fessura. I parametri $a_1, a_2, c_1, c_2, t_1, t_2$ sono costanti che determinano intensità,larghezza e posizione delle due fonti di luce. Nell'esperimento

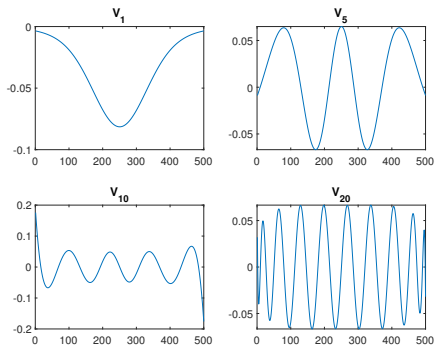
sono stati utilizzati i valori $a_1 = 2, a_2 = 1, c_1 = 6, c_2 = 2, t_1 = 0.8, t_2 = -5$.
Il kernel K e la soluzione f sono discretizzati con un campionamento $t_i = (i - 0.5)\pi/n,\ i = 1, ..., n$, per produrre una matrice A simmetrica di dimensione $n \times n$ e il vettore soluzione $\mathbf{x}^{\text{exact}}$. Dopo il corrispondente vettore $\mathbf{b}^{\text{exact}}$ è stato calcolato come $\mathbf{b}^{\text{exact}} = A\mathbf{x}^{\text{exact}}$. La funzione accetta come input la dimensione n e restituisce la matrice A, il vettore soluzione $\mathbf{x}^{\text{exact}}$ e il vettore $\mathbf{b}^{\text{exact}}$.
L'obbiettivo del problema è ottenere una buona approssimazione di $\mathbf{x}^{\text{exact}}$ a partire dalla conoscenza di A e del vettore $\mathbf{b}^{\text{exact}}$ a cui è stato aggiunto del rumore attraverso la funzione *randn*, pesata con un opportuno coefficiente *std*.

3.1.1 Analisi del problema

Il numero di condizione di A, di dimensione pari a $n = 500$, risulta uguale a $\text{cond}(A) = 1.72 \cdot 10^{20}$, il problema dunque è mal condizionato.
Poichè la dimensione della matrice A lo consente, è possibile calcolare facilmente la SVD e dall'analisi di questa possiamo vedere che i valori singolari di A decadono gradualmente a zero e i vettori singolari tendono ad avere più cambi di segno all'aumentare dell'indice i, come ci si aspettava da un problema discreto mal posto.



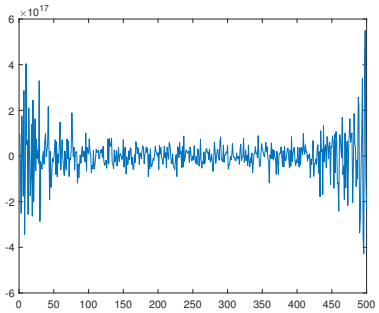
(a) Valori singolari



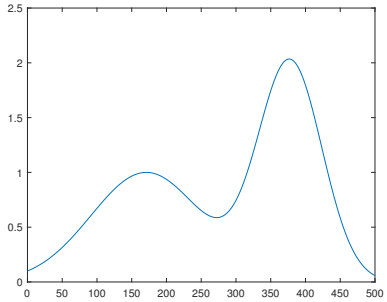
(b) Vettori singolari

3.1.2 Soluzione del problema

Il problema consiste nel determinare x dato il problema $A\mathbf{x} = \bar{\mathbf{b}}$ dove $\bar{\mathbf{b}} = \mathbf{b}^{\text{exact}} + \mathbf{err}$. Essendo il problema fortemente mal condizionato l'utilizzo di Gauss porta a un risultato completamente inutile con un errore pari a 10^{18} .



(c) Soluzione calcolata con Gauss

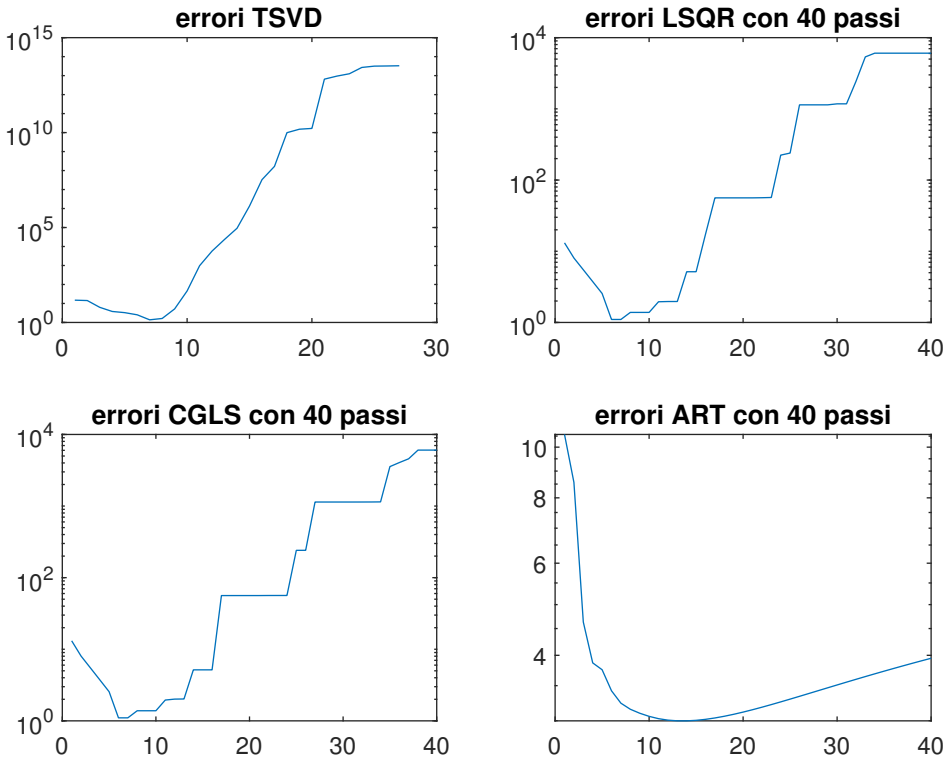


(d) Soluzione esatta

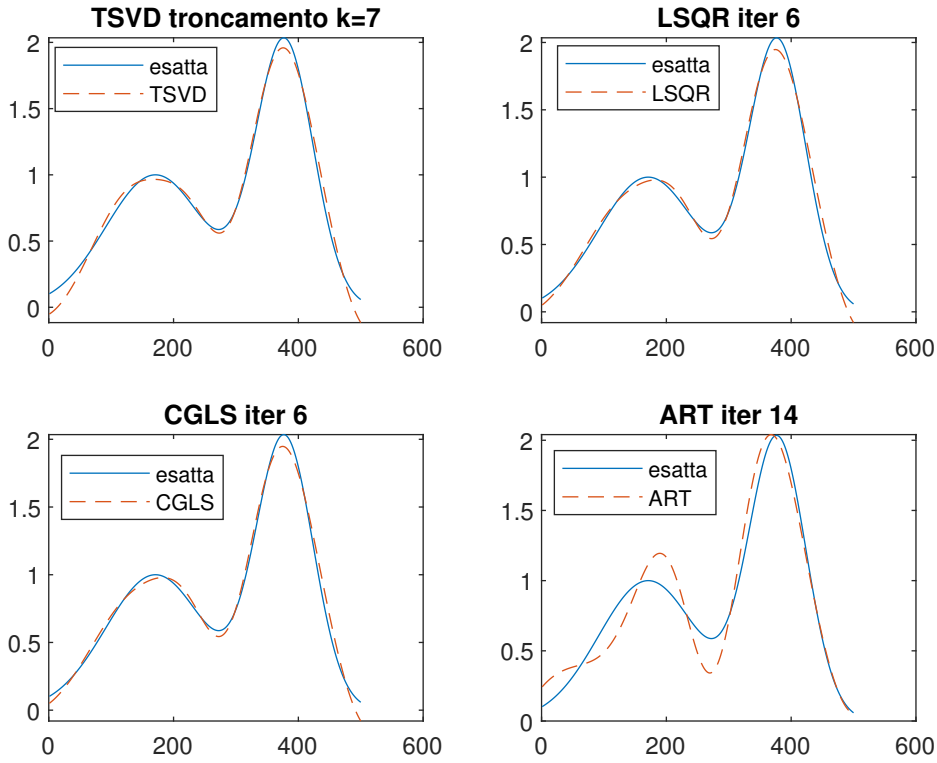
Sono stati utilizzati quindi i metodi di regolarizzazione TSVD, cgls, lsqr_b e art per la ricerca di una soluzione. Per tutti i metodi sono stati analizzati il tempo impiegato a calcolare la soluzione, l'errore minimo ottenuto e l'iterazione, o troncamento per quanto riguarda la TSVD, che ha fornito la soluzione migliore. Tutti i metodi iterativi sono stati calcolati utilizzando come stop 40 iterazioni, mentre la TSVD è stata analizzata sino a 20 passi più in avanti rispetto al parametro reg_parameter fornito dalla funzione l_curve. Ad ogni passo è stato calcolato l'errore assoluto $||\mathbf{x}^{(k)} - \mathbf{x}^{\text{exact}}||_2$ ed è stato rappresentato nel grafico in funzione di k, in tabella invece è rappresentato l'errore minimo relativo, calcolato come $\frac{||\mathbf{x}^{(migliore)} - \mathbf{x}^{\text{exact}}||_2}{||\mathbf{x}^{\text{exact}}||_2}$.
Sono stati fatti due test in cui si faceva variare il coefficiente moltiplicativo del rumore, nel primo test è stato usato std=10^{-2} mentre nel secondo è stato usato std=10^{-3}.

Test 1

metodo	tempo	migliore iterazione	errore minimo
TSVD	9.716000e-04	k=7	6.16%
lsqr_b	1.501110e-02	k=6	4.94%
cgls	7.962800e-03	k=6	4.94%
art	5.328640e-02	k=14	13%



(e) Errori TEST 1



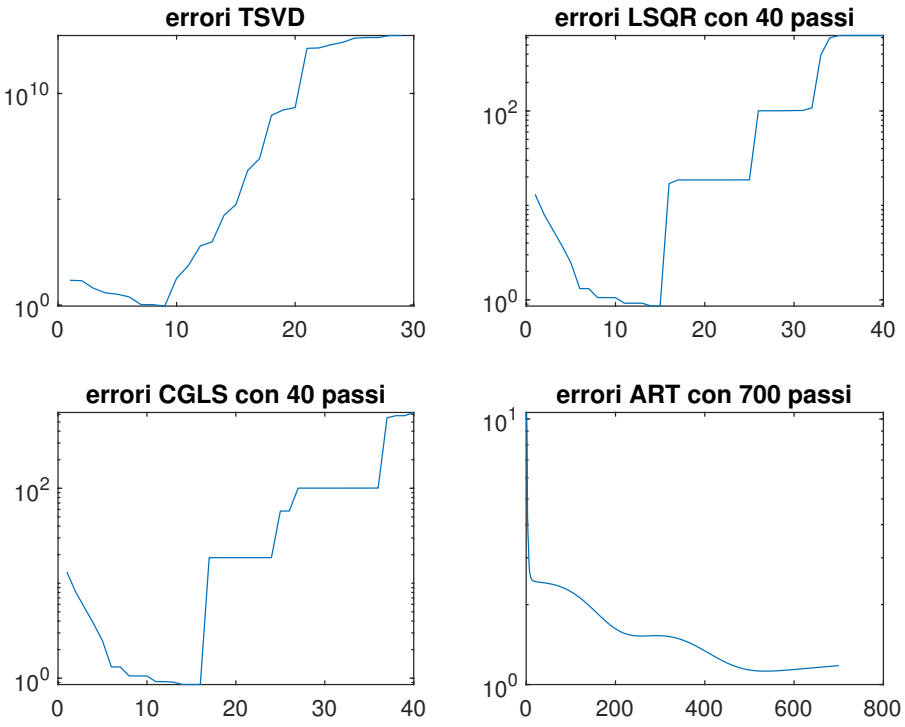
(f) Soluzioni migliori TEST 1

Test 2

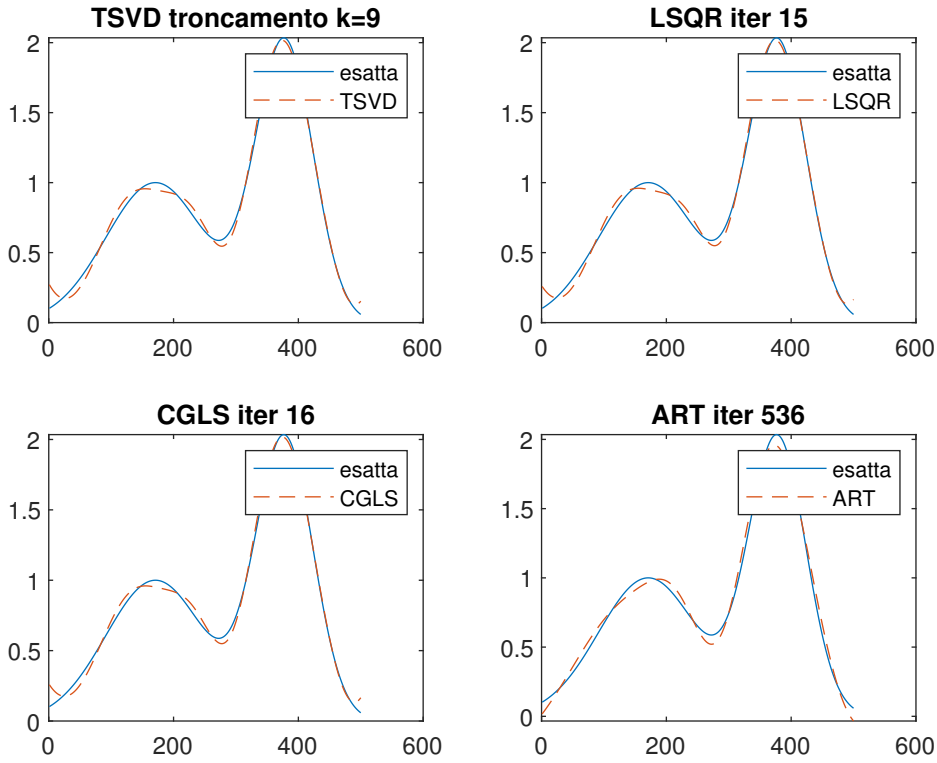
metodo	tempo	migliore iterazione	errore minimo
TSVD	1.158800e-03	k=9	4.04%
lsqr_b	5.281800e-03	k=15	3.85%
cgls	8.225300e-03	k=16	3.85%
art	7.118710e-02	k=40	10.83%

Poichè il risultato migliore di art coincide con il valore di stop è possibile aumentare il numero di iterazioni per ottenere un risultato migliore. L'errore minimo ottenibile è stato calcolato con stop a 700 passi.

metodo	tempo	migliore iterazione	errore minimo
art	1.149507e+00	k=536	5.03%



(g) Errori TEST 2



(h) Soluzioni migliori TEST 2

3.2 Problem test: gravity

Anche questo problema riguarda la discretizzazione di un equazione integrale di Fredholm del primo ordine

$$\int_0^1 K(s,t)f(t)dt = g(s), \qquad 0 < s < 1,$$

che rappresenta il modello matematico in una dimensione del rilevamento di gravità di una distribuzione di massa $f(t)$ collocata a una profondità d, dove la componente verticale di campo di gravità $g(s)$ è misurata sulla superficie.

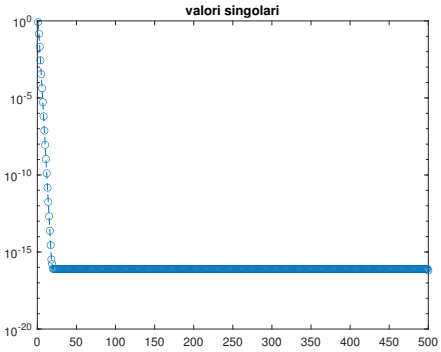
$$K(s,t) = d(d^2 + (s-t)^2)^{-3/2},$$

$$f(t) = \sin(\pi t) + 0.5\sin(2\pi t).$$

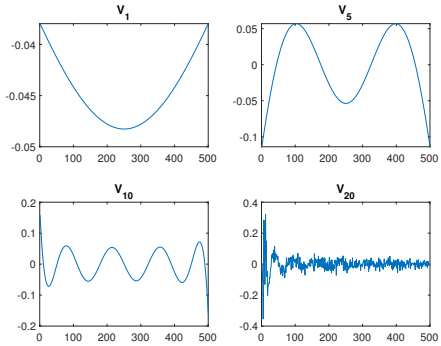
La funzione gravity, in maniera analoga alla funzione shaw, prende in input la dimensione n del sistema e restituisce la discretizzazione del kernel K come matrice A, il vettore $\mathbf{x}^{\text{exact}}$ e il vettore $\mathbf{b}^{\text{exact}}$ calcolato come $\mathbf{b}^{\text{exact}} = A\mathbf{x}^{\text{exact}}$. Come è stato fatto precedentemente, l'obbiettivo del problema è quello di calcolare una buona approssimazione di $\mathbf{x}^{\text{exact}}$, una volta aggiunto del rumore al vettore $\mathbf{b}^{\text{exact}}$.

3.2.1 Analisi del problema

Il numero di condizione di A risulta uguale a $\text{cond}(A) = 8 \cdot 10^{19}$, il problema è dunque mal condizionato. Eseguendo la SVD di A si può notare che anche in questo caso i valori singolari calano a zero gradualmente e i vettori singolari aumentano i cambi di segno all'aumentare dell'indice i.



(i) Valori singolari



(j) Vettori singolari

3.2.2 Soluzione del problema

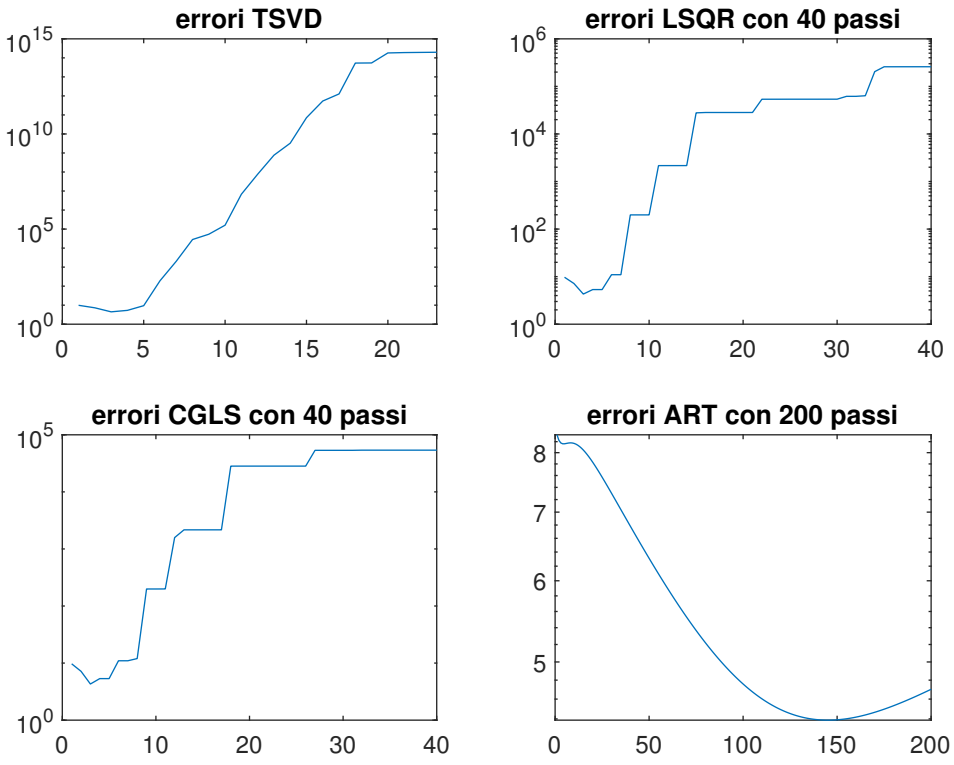
Anche in questo caso sono stati svolti due test facendo variare il parametro std associato al rumore e sono stati analizzati gli stessi valori del test shaw. Nel primo test è stato usato std=10^{-2} mentre nel secondo std=10^{-3}. In entrambi i test sono stati utilizzati $d = 0.25$, $n = 500$.

Test 1

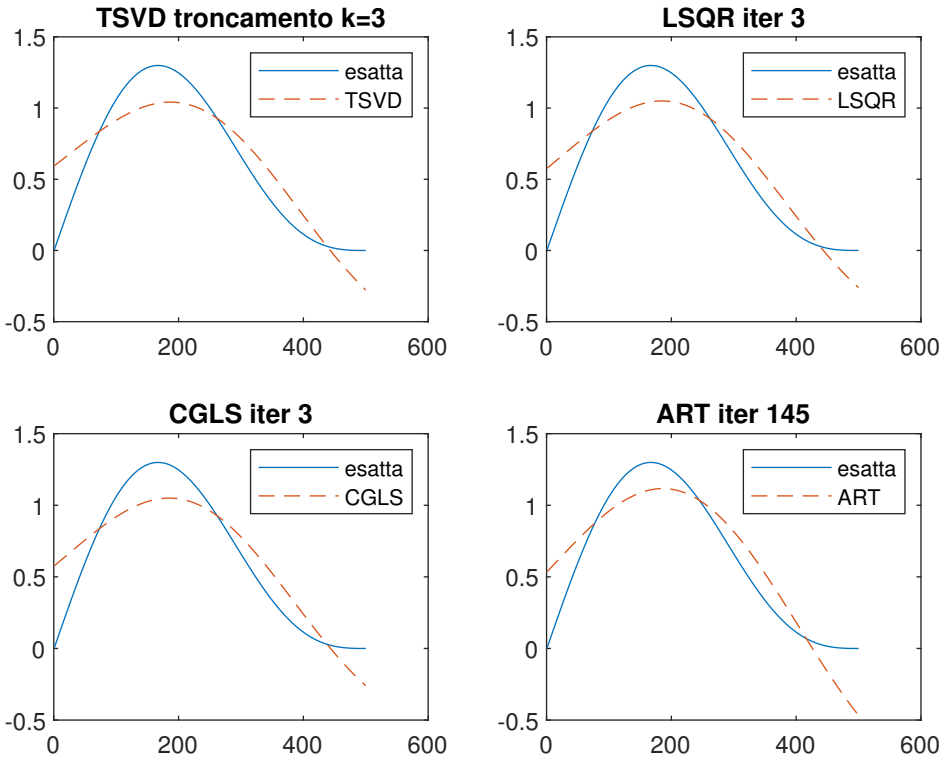
metodo	tempo	migliore iterazione	errore minimo
TSVD	1.285100e-03	k=3	25.23%
lsqr_b	1.137980e-02	k=3	24.32%
cgls	7.962800e-03	k=3	24.32%
art	7.074210e-02	k=40	38.37%

Poichè la migliore iterazione del metodo art coincide con il criterio di stop, possiamo aumentare il numero di iterazione per trovare una soluzione migliore. Si è trovato il miglior risultato impostando come stop a $k = 200$ ottenendo :

metodo	tempo	migliore iterazione	errore minimo
art	3.010428e-01	k=145	24.32%



(k) Errori TEST 1



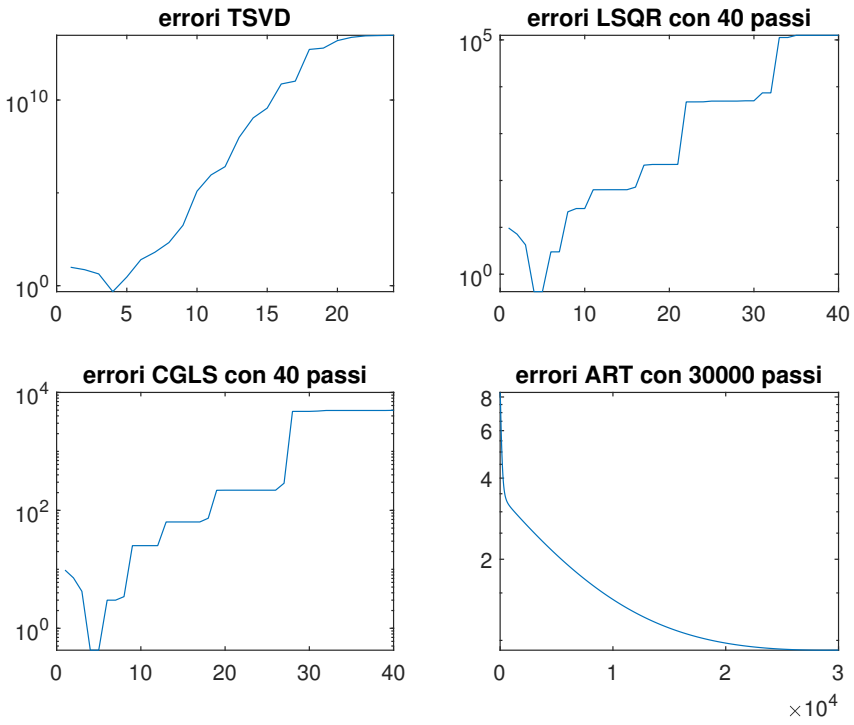
(l) Soluzioni migliori TEST 2

Test 2

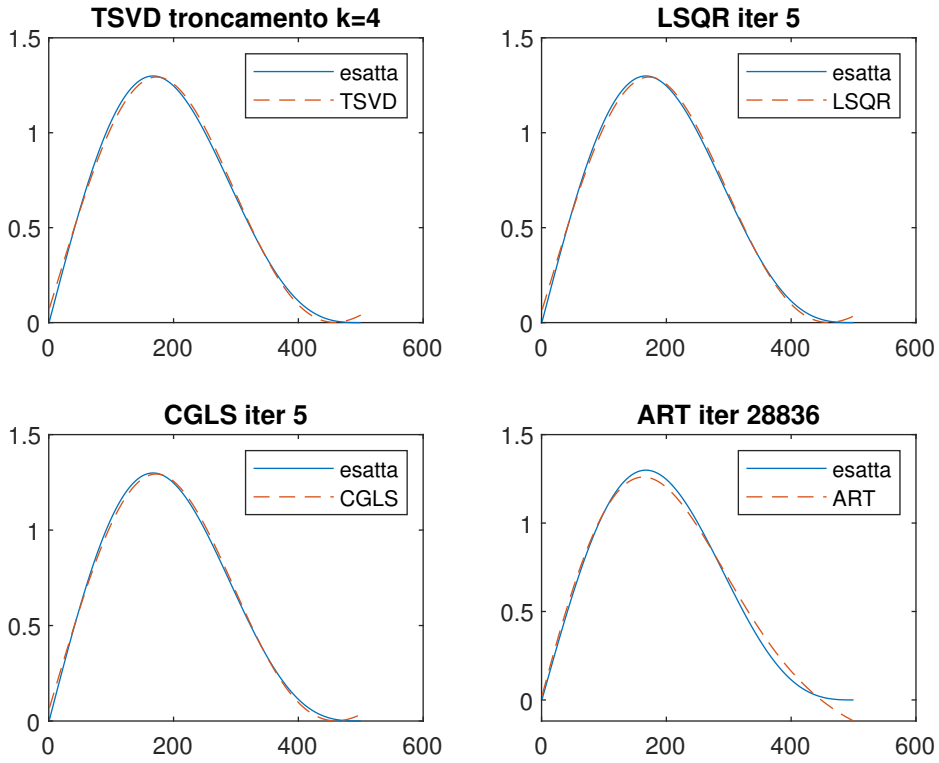
metodo	tempo	migliore iterazione	errore minimo
TSVD	2.219600e-03	k=4	2.71%
lsqr_b	1.875530e-02	k=5	2.4%
cgls	8.956500e-03	k=5	2.4%
art	7.646880e-02	k=40	41.58%

Come successo nel primo test, per migliorare il metodo art è necessario aumentare il numero di iterazioni. Dopo alcuni tentativi è stato trovato con criterio di stop k=30000 passi il risultato migliore.

metodo	tempo	migliore iterazione	errore minimo
art	3.914437e+01	k=28836	5.2%



(m) Errori TEST 2



(n) Soluzioni migliori TEST 2

3.3 Conclusioni

Se ci limitiamo all'analisi del solo valore di errore minimo trascurando il resto, possiamo dire che tutti e quattro i metodi portano a risultati abbastanza vicini tra loro, dove possiamo comunque evidenziare che lsqr_b e cgls, oltre a portare risultati pressoché uguali (come ci aspettavamo), portano anche i risultati migliori. La differenza sostanziale riguarda il tempo e il numero di iterazioni che portano alla soluzione migliore. Risulta evidente che il metodo art sia l'algoritmo meno conveniente, con dei tempi di convergenza e un numero di iterazioni necessarie a raggiungere il risultato migliore, nettamente superiori agli altri metodi. Considerando inoltre che per il calcolo della TSVD è stato necessario il calcolo della SVD, possiamo affermare che oltre a fornire i risultati migliori cgls e lsqr_b sono anche i metodi più veloci. Possiamo quindi dire che i metodi più convenienti sono lsqr_b e cgls.
Una caratteristica da sottolineare è che per tutti e quattro i metodi possiamo vedere (come ci aspettavamo) che all'aumentare delle iterazioni si raggiunge un risultato sempre più vicino a quello desiderato, fino a raggiungere il migliore possibile, una volta raggiunto il punto di minimo dell'errore continuare le iterazioni porta a un risultato che si allontana sempre di più da quello desiderato. Inoltre dai due test svolti per ciascun problema, possiamo vedere come l'effetto di una variazione dell'errore porti a risultati migliori a spese di un maggior numero di iterazioni.

Capitolo 4

Problemi bidimensionali

4.1 Immagini

Un immagine può essere definita come una funzione $f(x, y)$, dove x e y sono le coordinate spaziali, e il valore di f in ogni coordinata (x, y) è chiamata intensità o livello di grigio dell'immagine in quel punto. Quando (x, y) e il valore di f sono quantità discrete finite, chiamiamo l'immagine: **immagine digitale**.
Un immagine digitale a colori di dimensione $n \times m$ è rappresentata nel calcolatore attraverso tre matrici(RGB) di dimensione $n \times m$. Attraverso la sintesi additiva dei tre colori fondamentali si ricostruisce l'immagine con i colori originali.

(a) Matrice R (b) Matrice G (c) Matrice B

Figura 4.1: Scomposizione nelle 3 matrici RGB

Per immagini monocromatiche si memorizza un'unica matrice in cui il valore di ciascun elemento indica l'intensità del grigio che va da nero(valore minimo) a bianco(valore massimo).

Figura 4.2: Ricostruzione dell'immagine a colori

4.2 Acquisizione immagini

Il processo di acquisizione di un immagine può essere modellizzato attraverso una convoluzione del tipo

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} K(x-z,y-t)f(z,t)dzdt = g(x,y),$$

dove:

- $f(x,y)$=immagine originale
- $K(x,y)$=risposta impulsiva del sistema
- $g(x,y)$=immagine ottenuta

Possiamo applicare le proprietà che abbiamo visto quando abbiamo parlato delle matrici di Toeplitz, quindi trasformiamo la convoluzione in due dimensioni in un prodotto matrice per vettore

$$K\mathbf{f} = \mathbf{g},$$

dove K è la BTTB associata alla risposta impulsiva $K(x,y)$ mentre $\mathbf{f}$ e $\mathbf{g}$ sono i vettori colonna che rappresentano le matrici dell'immagine originale e dell'immagine ottenuta.

4.3 PSF

Nell'ambiente di elaborazione dell immagini ci si riferisce alla risposta impulsiva del sistema che acquisisce l'immagine con il termine **Point Spread Function**(PSF).

Questa descrive il modo in cui il sistema risponde a una sorgente luminosa puntiforme.

Un sistema che sfuoca un'immagine sarà descritto da una PSF gaussiana del tipo :

$$k(x,y) = \frac{\pi\sigma}{2} e^{-\sigma\frac{x^2+y^2}{2}}.$$

(a) originale (b) $\sigma = 0.3$ (c) $\sigma = 0.08$ (d) $\sigma = 0.03$

Figura 4.3: Sfuocatura con diversi valori di σ

4.4 Deblurring

Il problema del deblurring consiste nel ricostruire l'immagine originale a partire dall'immagine ottenuta che risulta sfuocata e anche affetta da errore. Utilizzando il modello discreto di convoluzione ciò equivale a risolvere il sistema

$$K\mathbf{f} = \mathbf{g}.$$

La difficoltà relativa a questi problemi risiede in particolare nella matrice BTTB K associata alla PSF discretizzata, questa infatti risulta :

- fortemente mal condizionata
- dimensione molto grande, infatti se se l'immagine f ha dimensione $m \times n$ allora K avrà dimensione pari a $mn \times mn$. Per esempio per un immagine 128×128 la matrice K avrà dimensione 16384×16384 con un numero di elementi pari a $2.6 \cdot 10^8$ e con un occupazione di memoria di 2.14 GB.

É quindi evidente che lavorare con la matrice K in forma completa è molto difficile, se non addirittura impossibile per immagini più grandi di 128×128,

a causa dell'elevatissimo utilizzo di memoria e della lentezza nei calcoli.

Fortunatamente grazie alla struttura di K è possibile ridurre l'occupazione di memoria e anche velocizzare i calcoli, rispettivamente grazie alla sola memorizzazione dello gnomone associato alla BTTB e alle proprietà legate alla FFT. Queste caratteristiche possono essere sfruttate esclusivamente per i metodi iterativi, che accedono alla matrice solo per effettuare prodotti matrice per vettore, per i metodi diretti invece è necessario lavorare con la matrice in forma completa e quindi non è possibile sfruttare tali proprietà.

Per gli esperimenti numerici sono state usate le tre immagini in figura le cui dimensioni sono state cambiate attraverso il software gimp. Gli esperimenti consistevano nel sfuocare le immagini tramite una funzione sfuoca, che prendeva in input l'immagine originale e il sigma della gaussiana relativa alla PSF, una volta ottenuta l 'immagine sfuocata a questa è stata aggiunta del rumore calcolato con la funzione randn pesata con un opportuno coefficiente std. Infine a partire da quest'ultima immagine, è stata ricostruita l'immagine originale attraverso un metodo di regolarizzazione.

(a) lena

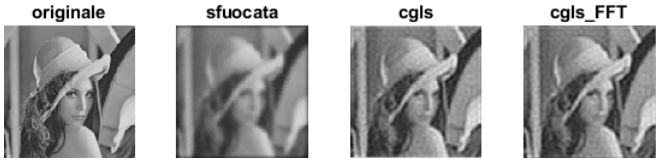
(b) satellite

(c) phantom

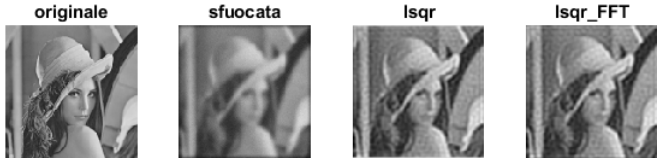
4.5 Confronti tra metodi ottimizzati e non ottimizzati

Come detto precedentemente utilizzare la matrice completa comporta un occupazione di memoria non trascurabile e maggiore lentezza nei calcoli, per questo sono stati ottimizzati gli algoritmi iterativi cgls, lsqr_b attraverso una

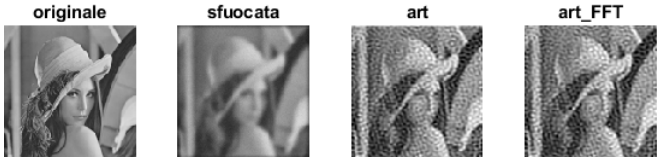
funzione tpax2 che esegue le moltiplicazioni matrici vettore attraverso la FFT utilizzando invece della matrice completa, lo gnomone associato a tale matrice. mentre il metodo art è stato ottimizzato, estraendo ad ogni passo la riga relativa alla matrice completa dallo gnomone. Per fare un confronto tra metodi originali e ottimizzati, è stata sfuocata l'immagine di Lena 128×128 con $\sigma = 0.3$, in seguito è stato aggiunto un rumore pesato con std=10^{-2} infine sono stati applicati metodi di regolarizzazione iterativi con stop a $k = 100$ per ricostruire l'immagine originale.



	tempo	errore minimo	iterazione migliore
cgls	2.935987e+01	9.4%	14
cgls_FFT	1.203646e+00	9.4%	14



	tempo	errore minimo	iterazione migliore
lsqr_b	2.938648e+01	9.4%	14
lsqr_b_FFT	1.090480e+00	9.4%	14



	tempo	errore minimo	iterazione migliore
art	1.337110e+02	18.26%	32
art_FFT	2.075727e+02	18.26%	32

	dimensione	elementi	occupazione di memoria
Matrice completa	16384x16384	2.684354e+8	2.14 GB
Gnomone	255x255	6.5025e+4	520 KB

Come si può vedere dai dati, l'ottimizzazione oltre a portare una drastica riduzione della memoria occupata porta anche a una riduzione dei tempi di computazione. L' unico metodo che risulta più lento è art, questo poichè nella forma originale ad ogni passo di iterazione viene estratta una riga dalla matrice completa per effettuare un prodotto tra vettori, mentre l'estrazione della riga in art_FFT viene eseguita in più passi, poichè dobbiamo estrarre ogni riga dallo gnomone e questo comporta maggior tempo. Nonostante ciò il metodo art classico non è possibile usarlo per matrici di dimensioni superiori a 128×128, a causa della dimensione di K, perciò anche se art_FFT risulta più lento di art, è necessario usarlo per immagini di grandi dimensioni. Un altro fatto da notare è che l'utilizzo della FFT porta a dei risultati pressochè uguali a quelli calcolati con i metodi classici, con una differenza tra i risultati dell'ordine di 10^{-16}.

4.6 Confronti tra i metodi

Come detto precedentemente, per la sperimentazione numerica sono stati confrontati i tre metodi iterativi: cgls, lsqr_b e art facendo variare le dimensioni dell immagini, il valore di sigma e il coefficiente std che moltiplica la componente associata al rumore. I parametri che sono stati confrontati sono: errore minimo, iterazione per cui si raggiungeva la soluzione migliore e il tempo impiegato a calcolare tutti i passi.

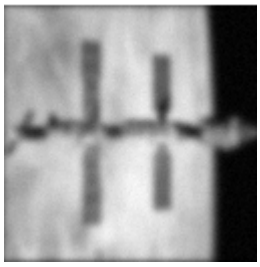
4.6.1 Immagini 128x128

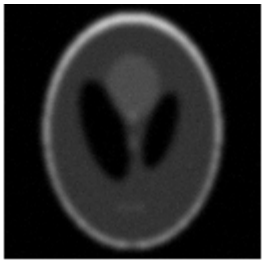
Sono stati effettuati quattro test, per tutti è stato usato come criterio di stop $k = 100$, tranne per il TEST 4 relativo all'immagine phantom, dove è stato usato k=150, poiché tutti e tre i metodi avevano come risultato migliore, quello relativo a k=100.

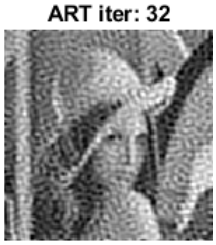
	TEST 1	TEST 2	TEST 3	TEST 4
std	10^{-2}	10^{-2}	10^{-2}	10^{-3}
sigma	0.3	0.07	0.6	0.3

Test1

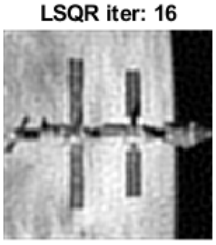
(d)

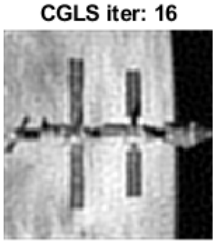


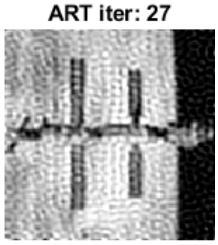




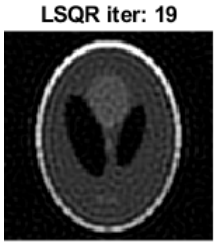
metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.154633e+00	k=14	9.40%
cgls	1.197831e+00	k=14	9.40%
art	2.493825e+02	k=32	18.26%

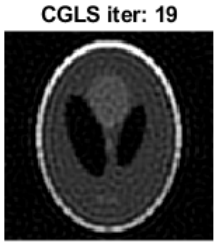


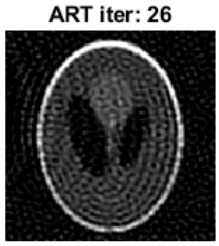




metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.133472e+00	k=16	8.05%
cgls	1.175647e+00	k=16	8.05%
art	2.159067e+02	k=27	14.14%

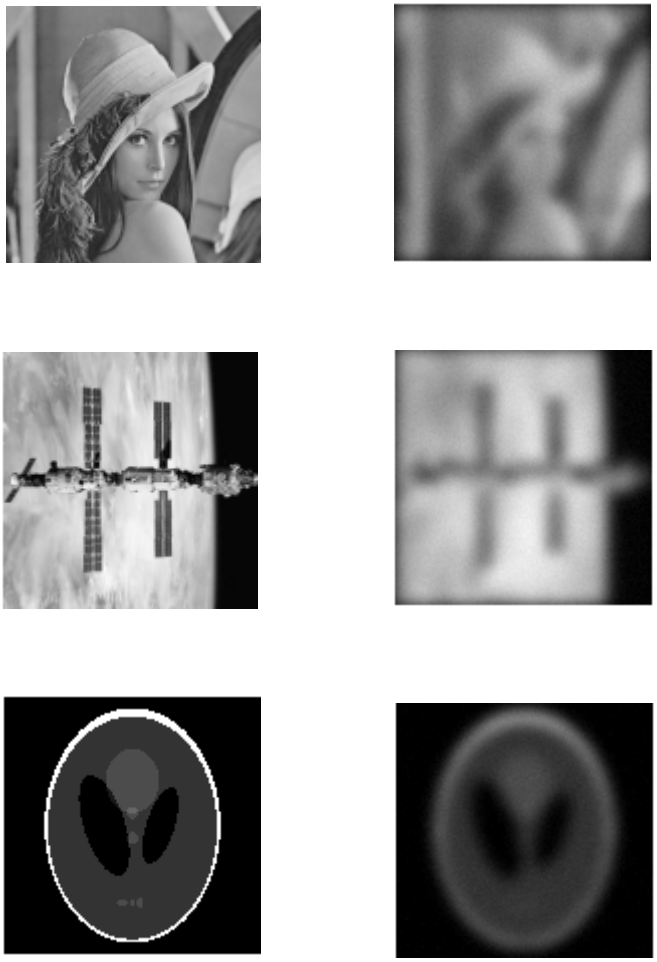




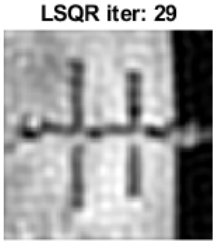


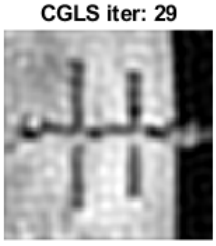
metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.117765e+00	k=19	59.13%
cgls	1.121868e+00	k=19	59.13%
art	2.135539e+02	k=26	74.54%

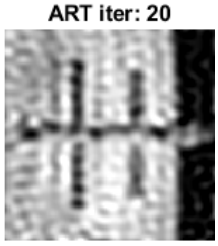
Test 2



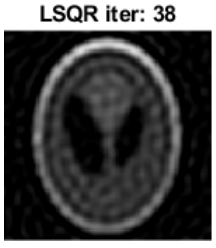
metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.021637e+00	k=26	14.26%
cgls	1.073633e+00	k=26	14.26%
art	2.000528e+02	k=23	21.14%

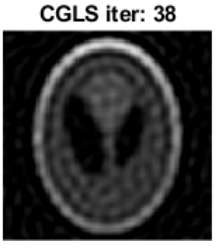


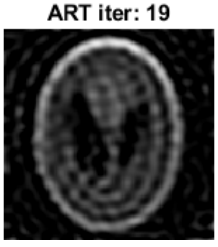




metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.034893e+00	k=29	12.33%
cgls	1.085636e+00	k=29	12.33%
art	2.395328e+02	k=20	16.72%



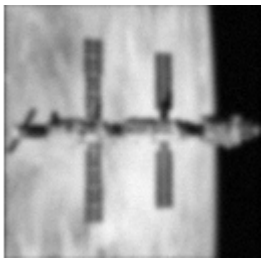


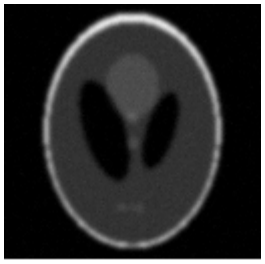


metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.111449e+00	k=38	83.06%
cgls	1.190239e+00	k=38	83.06%
art	2.126855e+02	k=19	92.65%

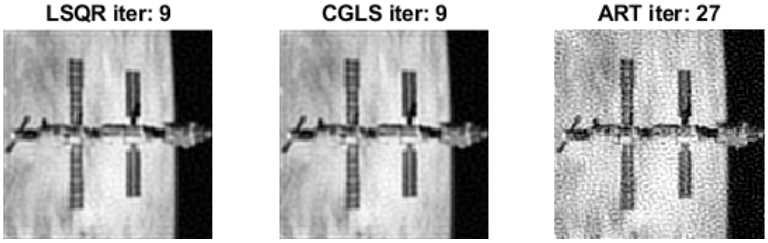
Test 3



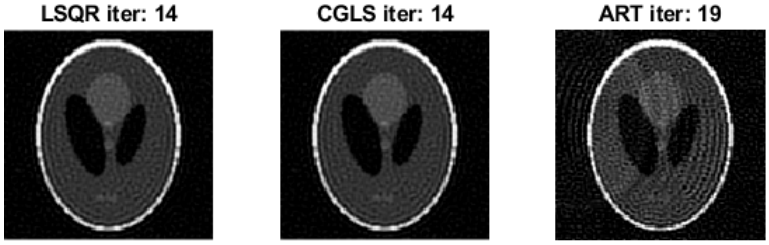





metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.017498e+00	k=9	7.61%
cgls	1.150968e+00	k=9	7.61%
art	2.185695e+02	k=27	16.96%

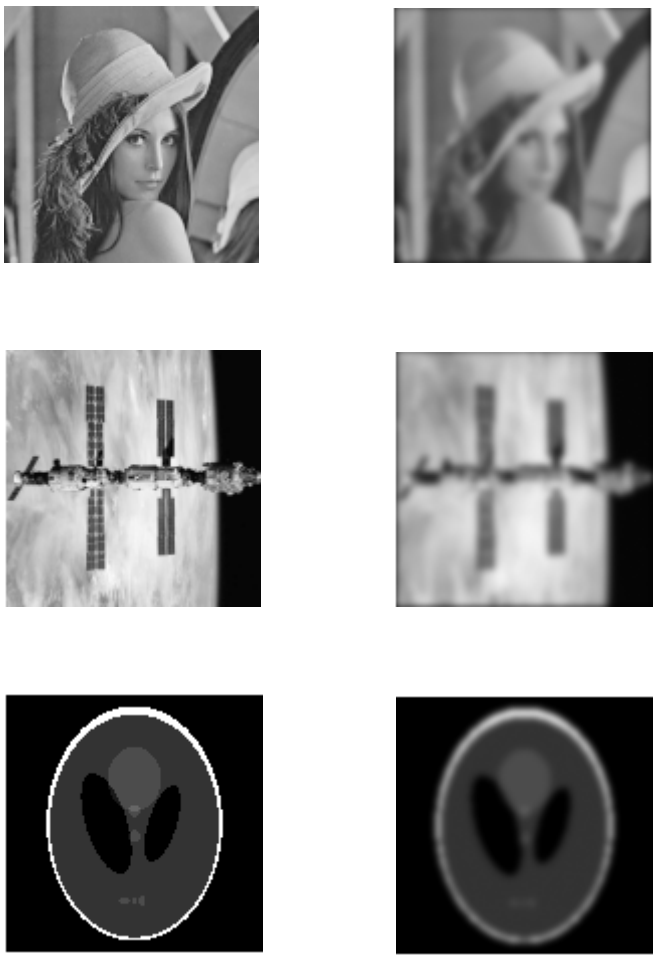


metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.078625e+00	k=9	6.39%
cgls	1.098181e+00	k=9	6.39%
art	2.323032e+02	k=27	13.37%

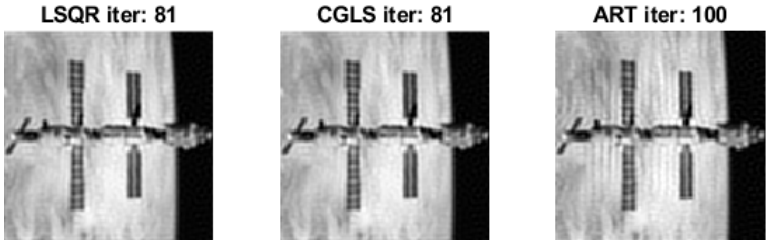


metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.107867e+00	k=14	48.05%
cgls	1.111388e+00	k=14	48.05%
art	2.167516e+02	k=19	62.28%

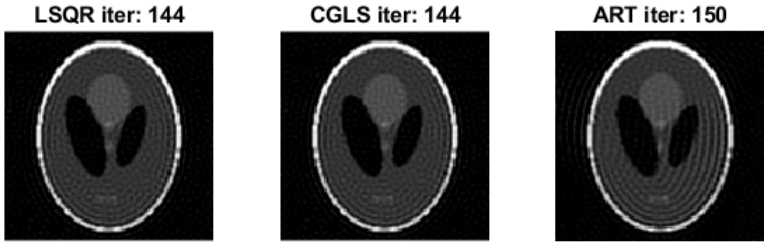
Test 4



metodo	tempo	migliore iterazione	errore minimo
lsqr_b	6.603475e-01	k=75	7.02%
cgls	6.573930e-01	k=75	7.02%
art	1.140171e+02	k=100	8.01%



metodo	tempo	migliore iterazione	errore minimo
lsqr_b	6.521009e-01	k=81	5.91%
cgls	6.527791e-01	k=81	5.91%
art	1.134186e+02	k=100	6.84%



metodo	tempo	migliore iterazione	errore minimo
lsqr_b	9.481113e-01	k=144	48.21%
cgls	9.754040e-01	k=144	48.21%
art	1.136006e+02	k=150	53.81%

4.6.2 Immagini 512x512

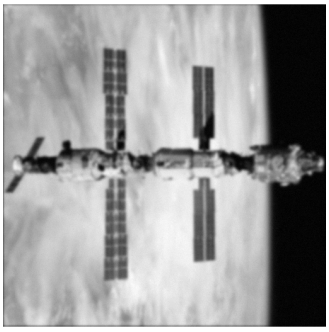
Analogamente alle immagini 128 × 128 sono stati effettuati quattro test facendo variare gli stessi parametri e analizzando gli stessi valori

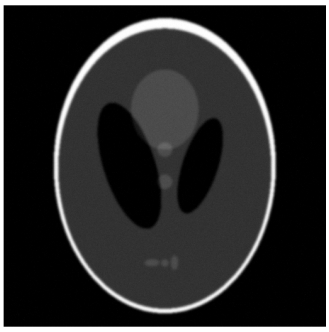
	TEST 1	TEST 2	TEST 3	TEST 4
std	10^{-2}	10^{-2}	10^{-2}	10^{-3}
sigma	0.3	0.07	0.6	0.3

A differenza dei test effettuati sulle immagini 128x128, per le immagini 512x512 sono stati utilizzati per l'analisi solamente i metodi cgls e lsqr_b. Il motivo di tale scelta riguarda il tempo di calcolo impiegato dal metodo art_FFT necessario per l'analisi di tali immagini. Il metodo come detto precedentemente risulta più lento dell' originale, ma necessario per operare su immagini grandi a causa dell'elevata occupazione di memoria della BTTB, allo stesso tempo però il metodo nonostante porti ad una soluzione accettabile, impiega troppo tempo ed è stato quindi scartato.

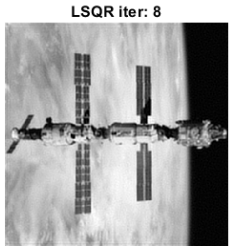
Test 1

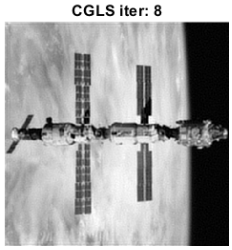


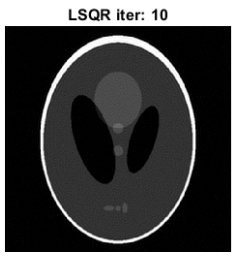


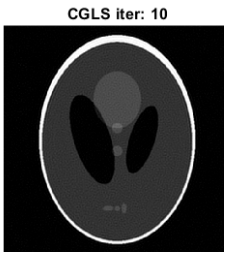
metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.731511e+01	k=9	5.60 %
cgls	1.684316e+01	k=9	5.60%





metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.726577e+01	k=8	3.15%
cgls	1.688175e+01	k=8	3.15%

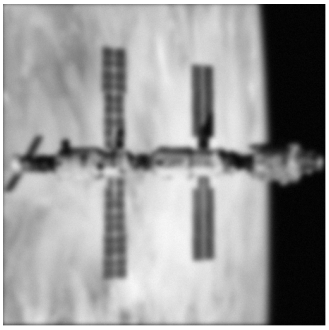


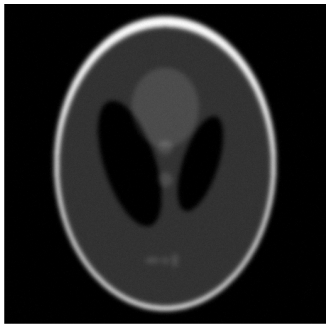


metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.728323e+01	k=10	29.77%
cgls	1.699006e+01	k=10	29.77%

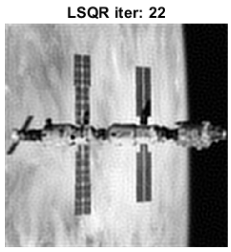
Test 2

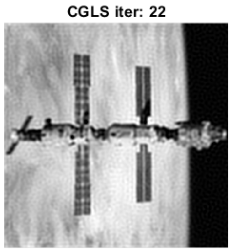


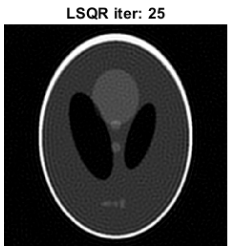


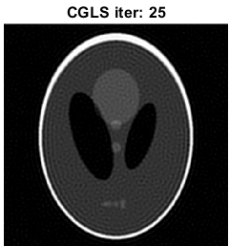
metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.802077e+01	k=20	8.37%
cgls	1.795190e+01	k=20	8.37%





metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.789952e+01	k=22	5.64%
cgls	1.780755e+01	k=22	5.64%

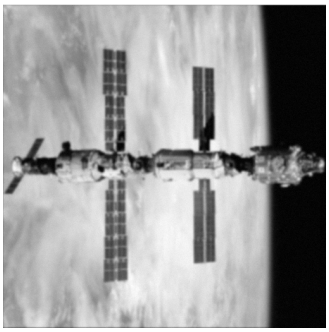


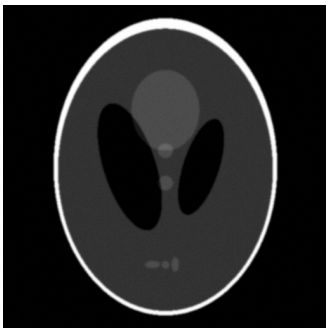


metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.807689e+01	k=25	37.69%
cgls	1.784216e+01	k=25	37.69%

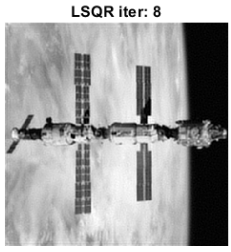
Test 3

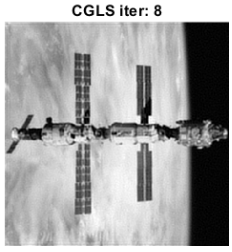


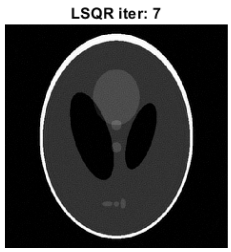


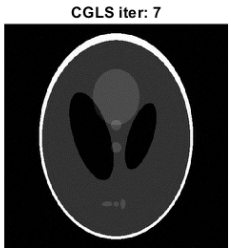
metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.754253e+01	k=6	4.57%
cgls	1.709765e+01	k=6	4.57%





metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.800234e+01	k=5	2.48%
cgls	1.787068e+01	k=5	2.48%





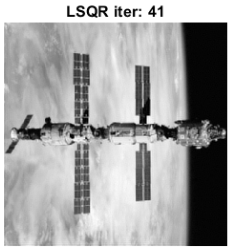
metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.749223e+01	k=7	25.16%
cgls	1.751124e+01	k=7	25.16%

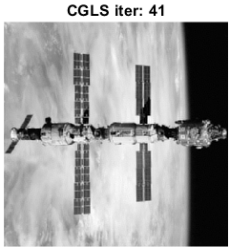
Test 4



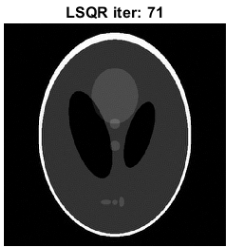


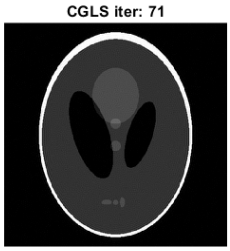

metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.765183e+01	k=50	3.87%
cgls	1.726185e+01	k=50	3.87%





metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.820040e+01	k=41	1.61%
cgls	1.824620e+01	k=41	1.61%





metodo	tempo	migliore iterazione	errore minimo
lsqr_b	1.754133e+01	k=71	23.99%
cgls	1.731664e+01	k=71	23.99%

4.7 Conclusioni

Possiamo concludere che i metodi migliori sono cgls e lsqr_b, sia per quanto riguarda il tempo impiegato a trovare la soluzione migliore sia per quanto riguarda l'errore relativo a tale soluzione, al contrario il metodo art si è dimostrato abbastanza inefficace a causa della lentezza nell'arrivare alla soluzione migliore, talmente elevata da rendere impossibile il suo utilizzo per le immagini 512x512. Dai test emerge che la qualità della soluzione trovata dipende sia dalla sfocatura dell'immagine sia dall'errore aggiunto alle immagini, in particolare possiamo notare che al diminuire di questi due parametri aumenta la qualità della soluzione trovata, inoltre possiamo vedere che una diminuzione della sfocatura porta a una maggiore velocità di convergenza del metodo mentre una diminuzione dell'errore porta a un risultato migliore a fronte di un maggior numero di iterazioni. Possiamo inoltre notare che all'aumentare della dimensione dell'immagine, aumenta la qualità della soluzione trovata. In conclusione possiamo affermare che l'ottimizzazione dei metodi lsqr_b e cgls è riuscita con successo mentre non possiamo affermare lo stesso riguardo l'ottimizzazione del metodo art , questo infatti nonostante in teoria riesca a lavorare con matrici di grandi dimensioni, in pratica si è dimostrato inefficiente a causa del tempo impiegato a estrarre ogni riga dalla matrice dello gnomone.

Bibliografia

[1] G.H. Golub and C.F. Van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, third edition, 1996.

[2] M.T. Galizia. *Analisi di Fourier con elementi di trasformate di Laplace*. Progetto Leonardo, Bologna,2001.

[3] M. Redivo-Zaglia,G. Rodriguez. *smt: a Matlab toolbox for structured matrices*. Numerical Algorithms, 59(4),2012, 639-659.

[4] G. Rodriguez, S. Seatzu. *Introduzione alla matematica Applicata e Computazionale*. Pitagora Editrice, Bologna,2010.

[5] P.C. Hansen. *Regularization tools version 4.0 for Matlab 7.3*. Numerical algorithms, 46(2),2007, 189-194.

[6] P.C. Hansen. *Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion*. Society for Industrial and Applied Mathematics,Philadelfia,1998.

[7] R.C. Gonzalez, R.E. Woods. *Digital Image Processing* .Digital Prentice hall Upper Saddle River,New Jersey,2002.